OSP-9705

US

JC813 U.S. PTO 09/716378 11/21/00

日 本 国 特 許 庁
PATENT OFFICE
JAPANESE GOVERNMENT

#3

別紙添付の書類に記載されている事項は下記の出願書類に記載されてる事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed in this Office.

出願年月日
Date of Application: １９９９年１１月２２日

出願番号
Application Number: 平成１１年特許願第３３１９６０号

出願人
Applicant(s): 日本電気株式会社

CERTIFIED COPY OF PRIORITY DOCUMENT

２０００年 ７月２１日

特許庁長官
Commissioner, Patent Office 及川耕造

出証番号 出証特２０００－３０５８１６０

【書類名】　特許願
【整理番号】　74310321
【提出日】　平成11年11月22日
【あて先】　特許庁長官　殿
【国際特許分類】　G06F　9/00
【発明の名称】　マイクロプロセッサシステム
【請求項の数】　11
【発明者】
　【住所又は居所】　東京都港区芝五丁目７番１号　日本電気株式会社内
　【氏名】　峰　一雅
【特許出願人】
　【識別番号】　000004237
　【氏名又は名称】　日本電気株式会社
【代理人】
　【識別番号】　100108578
　【弁理士】
　【氏名又は名称】　高橋　詔男
【代理人】
　【識別番号】　100064908
　【弁理士】
　【氏名又は名称】　志賀　正武
【選任した代理人】
　【識別番号】　100101465
　【弁理士】
　【氏名又は名称】　青山　正和
【選任した代理人】
　【識別番号】　100108453
　【弁理士】
　【氏名又は名称】　村山　靖彦

【手数料の表示】

【予納台帳番号】　008707

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　9709418

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　マイクロプロセッサシステム

【特許請求の範囲】

【請求項１】　プログラムに記述された命令を実行するマイクロプロセッサシステムであって、

第１の命令セットをハードウェア上で実行すると共に第２の命令セットをソフトウェア上で実行するメインプロセッサと、

前記メインプロセッサの管理下で動作して前記第２の命令セットを自発的にフェッチしてハードウェア上で実行するコプロセッサと、

を備えたことを特徴とするマイクロプロセッサシステム。

【請求項２】　前記コプロセッサは、

前記第２の命令セットのうち、前記メインプロセッサの管理下にあるデータを操作する必要を生ずる特定の命令に遭遇した場合、前記メインプロセッサに通知を発行して当該命令の実行を依頼することを特徴とする請求項１に記載されたマイクロプロセッサシステム。

【請求項３】　前記コプロセッサは、

前記第２の命令セットのうち、実行頻度の高い所定の複数の命令に対して予め割り付けられた専用割り込みベクタを用いて前記通知を発行することを特徴とする請求項２に記載されたマイクロプロセッサシステム。

【請求項４】　前記コプロセッサは、

前記第２の命令セットに属する命令を実行する過程で発生するデータを保持するためのスタックメモリと、該スタックメモリ内の最新データのアドレスを保持するスタックポインタとを有し、前記特定の命令を実行する過程で発生する処理のうち、前記スタックメモリのスタックポインタを更新するための処理を実行するハードウエア資源を備えたことを特徴とする請求項２に記載されたマイクロプロセッサシステム。

【請求項５】　前記コプロセッサは、

現在処理中の前記第２の命令セットに属する命令のアドレスを保持するためのプログラムカウンタを有し、前記特定の命令を実行する過程で発生する処理のう

ち、前記プログラムカウンタを更新するための処理を実行するハードウエア資源を備えたことを特徴とする請求項２に記載されたマイクロプロセッサシステム。

【請求項６】　前記コプロセッサは、

前記通知を行う必要が発生したことを表す情報を保持するための状態レジスタを有し、

前記メインプロセッサは、

前記状態レジスタを定期的にアクセスして、該状態レジスタの内容から前記コプロセッサが前記特定の命令に遭遇したことを把握して、前記特定の命令を実行することを特徴とする請求項２に記載されたマイクロプロセッサシステム。

【請求項７】　前記メインプロセッサは、

前記コプロセッサからの前記専用割り込みベクタをエンコードして、処理すべき前記特定の命令に対応する割り込みハンドラを特定する割り込み要求受付回路を備えたことを特徴とする請求項２に記載されたマイクロプロセッサシステム。

【請求項８】　前記メインプロセッサは、

前記コプロセッサの命令キューを参照して、処理すべき前記特定の命令に対応する割り込みハンドラを特定することを特徴とする請求項２に記載されたマイクロプロセッサシステム。

【請求項９】　前記コプロセッサは、

スタックアーキテクチャを有することを特徴とする請求項１ないし８の何れかに記載されたマイクロプロセッサシステム。

【請求項１０】　前記コプロセッサは、

スタックデータのうち、先頭側の所定数のデータを保持するスタックトップレジスタと、

外部に設けられたスタックメモリと前記スタックトップレジスタとの間に設けられ、前記スタックメモリに保持されたデータの一部をキャッシュするキャッシュメモリと、

を含むことを特徴とする請求項９に記載されたマイクロプロセッサシステム。

【請求項１１】　プログラムに記述された複数の処理内容に対応づけて、複数の前記コプロセッサを備えたことを特徴とする請求項１ないし１０の何れかに

記載されたマイクロプロセッサシステム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

この発明は、プログラムに記述された命令を実行するマイクロプロセッサに関し、特にメインプロセッサとコプロセッサとから構成されるマイクロプロセッサシステムに関する。

【０００２】

【従来の技術】

従来、各種の命令セットを実行するための汎用のマイクロプロセッサがある。この汎用のマイクロプロセッサは、ハードウェア上での処理に加えて、ソフトウェア上で適応的に処理する機能を有しているので、Ｊａｖａなどインタプリタ言語を実行するための命令セットや、他のＣＰＵをエミュレートするための命令セットなどのように、異なる体系の各種の命令セットに柔軟に対応することができる。

【０００３】

また、上述のような汎用のマイクロプロセッサをメインプロセッサとし、このメインプロセッサの管理下で動作するマイクロプロセッサをコプロセッサとして備えるマイクロプロセッサシステムがある。この種のマイクロプロセッサシステムでは、コプロセッサは、メインプロセッサの処理の一部を高速化するためのものものであって、例えばメインプロセッサ側のソフトウェア上で行われる特定の命令セットを実行するための専用のハードウェアを備えて構成されている。

【０００４】

【発明が解決しようとする課題】

ところで、上述の汎用のマイクロプロセッサによれば、柔軟に各種の命令セットを実行することができるものの、例えばＪａｖａ言語などの特定のインタプリタ言語の命令を実行する場合、処理に時間を要するという問題がある。つまり、汎用のマイクロプロセッサは、各種の命令セットに柔軟に対応する必要上、ハードウェアのアーキテクチャとして、汎用レジスタアーキテクチャや、アキュムレ

ータアーキテクチャを採用しているのが通例である。このため、ハードウェアとの親和性の観点では、Ｃ言語などの高級言語の処理に適している反面、Ｊａｖａ言語などのインタプリタ言語についてはソフトウェア上で対処する必要があり、したがって処理に時間を要していた。

【０００５】

また、メインプロセッサとコプロセッサとから構成された上述のマイクロプロセッサシステムによれば、Ｊａｖａ言語などの特定のプログラム言語の命令セットのすべての命令を専用に処理するものとしてコプロセッサのハードウェアを構成すれば、この特定の命令セットについては、専用のハードウェア上で実行される。このため、その特定の命令セットを高速に実行することができるものの、コプロセッサの回路規模が大きくなり、したがって全体としてチップ面積が増大するという問題がある。

【０００６】

この発明は、上記事情に鑑みてなされたもので、回路規模の増大を抑えつつ、特定の命令セットを高速に実行することが可能なマイクロプロセッサシステムを提供することを目的とする。

【０００７】

【課題を解決するための手段】

この発明は、上記課題を解決するため、以下の構成を有する。

すなわち、この発明にかかるマイクロプロセッサシステムは、プログラムに記述された命令を実行するマイクロプロセッサシステムであって、第１の命令セット（例えば後述するプログラムメモリ３００に格納されているプログラムの命令）をハードウェア上で実行すると共に第２の命令セット（例えば後述するプログラムメモリ４００に格納されているプログラムの命令）をソフトウェア上で実行するメインプロセッサ（例えば後述するメインプロセッサ１００に相当する構成要素）と、前記メインプロセッサの管理下で動作して前記第２の命令セットを自発的にフェッチしてハードウェア上で実行するコプロセッサ（例えば後述するコプロセッサ２００に相当する構成要素）と、を備えたことを特徴とする。

【０００８】

この構成によれば、第２の命令セットについては、メインプロセッサまたはコプロセッサにより適応的に処理することができる。したがって、例えば、原則として第２の命令セットの処理をコプロセッサのハードウェア上で処理することとし、例外として第２の命令セットの内のソフトウェアによる処理に適した命令のみをメインプロセッサで実行するようにすれば、マイクロプロセッサシステムに要求される処理能力に対して、全体としてハードウェアの規模を縮小される。したがって、回路規模の増大を抑えつつ、特定の命令セット（第２の命令セット）を高速に実行することが可能となる。

【０００９】

また、前記コプロセッサは、命令フェッチ回路とプログラムカウンタと命令キューとを有し、命令フェッチ回路が、プログラムカウンタの値を受け取り、この値をアドレスとして使用してプログラムメモリにアクセスし、得られたデータを命令キューにセットすることを特徴とする。

【００１０】

この構成によれば、コプロセッサが自発的に命令をフェッチして実行することができ、コプロセッサが、第２の命令セットをハードウェアで実行することができる。こうすることで、メインプロセッサがプログラムメモリから第２の命令セットをデータとしてフェッチし、コプロセッサの命令キューにセットする、という動作を実行せずにすみ、メインプロセッサの負担を軽減できるとともに、フェッチの動作に要する時間が短くなり、第２の命令の実行速度の向上が実現でき、システム全体の処理性能の向上が実現可能となる。

【００１１】

また、前記コプロセッサは、例えば、前記第２の命令セットのうち、前記メインプロセッサの管理下にあるデータ（例えば後述するデータメモリ５００に格納されているデータ）を操作する必要を生ずる特定の命令に遭遇した場合、前記メインプロセッサに通知を発行して当該命令の実行を依頼することを特徴とする。

【００１２】

この構成によれば、メインプロセッサの管理下にあるデータを操作する必要を生ずる特定の命令については、メインプロセッサのソフトウェア上で、このメイ

ンプロセッサの管理下にあるデータを操作して処理される。換言すれば、ソフトウェアによる処理に適した命令や、コプロセッサでは処理不能な命令については、メインプロセッサのソフトウェア上で処理される。これにより、メインプロセッサのソフトウエアが有効に活用されると共にコプロセッサのハードウェアの規模を必要最小限に抑えることが可能となり、したがって全体としてハードウェアの規模を有効に抑えることが可能となる。

【００１３】

さらに、前記コプロセッサは、例えば、前記第２の命令セットのうち、実行頻度の高い所定の複数の命令に対して予め割り付けられた専用割り込みベクタ（例えば後述する専用割り込み要求信号Ｓ２９０Ａ～Ｓ２９０Ｆにより指定される専用割り込みベクタに相当する要素）を用いて前記通知を発行することを特徴とする。

【００１４】

この構成によれば、第２の命令セットのうち、実行頻度の高い命令に遭遇した場合、この命令に割り付けられた専用割り込み要求信号を用いてメインプロセッサへの通知が行われる。メインプロセッサは、この専用割り込み要求信号により、コプロセッサからの通知に加えて、コプロセッサが遭遇した命令が把握される。したがって、メインプロセッサ側において、コプロセッサからの依頼により処理すべき命令を識別するための処理を省くことが可能となる。

【００１５】

さらにまた、前記コプロセッサは、例えば、前記第２の命令セットに属する命令を実行する過程で発生するデータを保持するためのスタックメモリ（例えば後述するスタックメモリ２７０に相当する構成要素）と、該スタックメモリ内の最新データのアドレスを保持するスタックポインタ（例えば後述するスタックポインタ２６０Ａに相当する構成要素）とを有し、前記特定の命令を実行する過程で発生する処理のうち、前記スタックメモリのスタックポインタを更新するための処理を実行するハードウエア資源（例えば後述するデコーダ２４０Ａ、処理手順制御信号発生回路２４０Ｂ、ＳＰ用増減値発生回路２４０Ｃ、加算器２５０Ａに相当する構成要素）を備えたことを特徴とする。

【００１６】

この構成によれば、メインプロセッサに処理を依頼する命令であっても、この命令を実行する過程で発生する処理のうち、スタックポインタを更新するための処理については、コプロセッサのハードウェア上で行われる。したがって、メインプロセッサが依頼を受けてこの命令を処理する過程において、スタックポインタを更新するための処理を省くことができる。このため、メインプロセッサのソフトウエア上での処理が簡略化されると共に、コプロセッサのハードウェアが有効に活用され、メインプロセッサでのソフトウェアによる処理時間を短縮することができる。

【００１７】

さらにまた、前記コプロセッサは、例えば、現在処理中の前記第２の命令セットに属する命令のアドレスを保持するためのプログラムカウンタ（例えば後述するプログラムカウンタ２６０Ｂに相当する構成要素）を有し、前記特定の命令を実行する過程で発生する処理のうち、前記プログラムカウンタを更新するための処理を実行するハードウエア資源（例えば後述するデコーダ２４０Ａ、処理手順制御信号発生回路２４０Ｂ、ＰＣ用増減値発生回路２４０Ｄ、加算器２５０Ｂに相当する構成要素）を備えたことを特徴とする。

【００１８】

この構成によれば、メインプロセッサに処理を依頼する命令であっても、この命令を実行する過程で発生する処理のうち、プログラムカウンタを更新するための処理については、コプロセッサのハードウェア上で行われる。したがって、メインプロセッサが依頼を受けてこの命令を処理する過程において、プログラムカウンタを更新するための処理を省くことができる。このため、メインプロセッサのソフトウエア上での処理が簡略化されると共に、コプロセッサのハードウェアが有効に活用され、メインプロセッサでのソフトウェアによる処理時間を短縮することができる。

【００１９】

さらにまた、前記コプロセッサは、例えば、前記通知を行う必要が発生したことを表す情報を保持するための状態レジスタを有し、前記メインプロセッサは、

前記状態レジスタを定期的にアクセスして、該状態レジスタの内容から前記コプロセッサが前記特定の命令に遭遇したことを把握して、前記特定の命令を実行することを特徴とする。

この構成によれば、メインプロセッサが定期的に状態レジスタをアクセスして、コプロセッサが特定の命令に遭遇したことを把握するので、メインプロセッサに対し、割り込み要求信号を用いて割り込み処理を行う必要がなくなる。したがって、コプロセッサからメインプロセッサへの通知を簡略化することが可能となる。

【００２０】

さらにまた、前記メインプロセッサは、例えば、前記コプロセッサからの前記専用割り込みベクタをエンコードして、処理すべき前記特定の命令に対応する割り込みハンドラを特定する割り込み要求受付回路（例えば後述する割り込み要求受付回路１９０に相当する構成要素）を備えたことを特徴とする。

この構成によれば、メインプロセッサにおいて、コプロセッサから入力する専用割り込みベクタに基づいて、直接的に割り込みハンドラが特定される。したがって、メインプロセッサ側で起動すべき割り込みハンドラを特定するためのソフトウエア上の処理が不要となり、メインプロセッサでの処理時間を短縮することが可能となる。

【００２１】

さらにまた、前記メインプロセッサは、例えば、前記コプロセッサの命令キューを参照して、処理すべき前記特定の命令に対応する割り込みハンドラを特定することを特徴とする。

この構成によれば、メインプロセッサにおいて、コプロセッサの命令キューを参照してコプロセッサが遭遇した命令が把握され、直接的に割り込みハンドラが特定される。したがって、メインプロセッサ側で起動すべき割り込みハンドラを特定するためのソフトウエア上の処理が簡略化され、メインプロセッサでの処理時間を短縮することが可能となる。

【００２２】

さらにまた、前記コプロセッサは、スタックアーキテクチャ（例えば後述する

コプロセッサ２００，２００Ａ，２００Ｂが採用するアーキテクチャに相当する要素）を有することを特徴とする。

この構成によれば、コプロセッサにおいて、サブルーチンの頻繁な使用を伴う命令や、ゼロオペランドのアドレッシングを伴う命令の処理が効率的に行われる。したがってＪａｖａなどのスタックアーキテクチャのインタプリタ言語の命令を効率的に実行することが可能となる。

【００２３】

さらにまた、スタックアーキテクチャを採用する前記コプロセッサは、例えば、スタックデータのうち、先頭側の所定数のデータを保持するスタックトップレジスタ（例えば後述するスタックトップレジスタ２７０Ａに相当する構成要素）と、外部に設けられたスタックメモリと前記スタックトップレジスタとの間に設けられ、前記スタックメモリに保持されたデータの一部をキャッシュするキャッシュメモリ（例えば後述すキャッシュメモリ２７０Ｃに相当する構成要素）と、を含むことを特徴とする。

この構成によれば、スタックデータのうち、先頭側の所定数のデータをアクセスする頻度の高いプログラム言語の命令を効率的に実行することができると共に、スタックメモリの容量を適応的に拡大することが可能となる。

【００２４】

さらにまた、プログラムに記述された複数の処理内容に対応づけて、複数の前記コプロセッサ（例えば後述するコプロセッサ２００Ｘ～２００Ｚに相当する構成要素）を備えたことを特徴とする。

この構成によれば、プログラム中の複数の処理を並列処理することが可能となり、処理を高速化することができる。

【００２５】

【発明の実施の形態】

次に、この発明の実施の形態について、図面を参照して詳細に説明する。

＜実施の形態１＞

図１に、この発明の実施の形態１にかかるマイクロプロセッサシステムの構成を示す。このマイクロプロセッサシステムは、異なるプログラム言語で記述され

た複数種類の命令セットを実行可能なように構成されたものであって、いわゆる汎用レジスタ・アーキテクチャを採用する汎用のマイクロプロセッサ１００と、スタック・アーキテクチャを採用するコプロセッサ２００とを主要な構成要素として備える。

【００２６】

メインプロセッサ１００は、Ｃ言語やアセンブラなどのプログラム言語で記述された第１の命令セットをハードウェア上で実行し、Ｊａｖａ言語などの特定のプログラム言語で記述された第２の命令セットをソフトウェア上で実行するものである。また、コプロセッサ２００は、メインプロセッサ１００の管理下で協調動作するものであって、上述の第１の命令セットとは異なるＪａｖａなどの第２の命令セットをハードウェア上で実行するものである。

【００２７】

プログラムメモリ３００は、メインプロセッサ１００の一連の動作を制御するためのプログラムであって第１の命令セットの機械語を格納する。プログラムメモリ４００は、Ｊａｖａなどの第２の命令セットの中間コードを格納する。データメモリ５００は、メインプロセッサ１００に入力される演算データやその演算結果など、メインプロセッサ１００で操作されるデータを格納する。

この実施の形態１では、プログラムメモリ３００には、Ｃ言語で記述されたプログラムの機械語であってメインプロセッサの動作手順が記述された機械語が展開されているものとし、プログラムメモリ４００には、Ｊａｖａ言語などのインタプリタ言語の中間コードが展開されているものとする。

【００２８】

これらプログラムメモリ３００，４００、およびデータメモリ５００は、外部バス６００を介して互いに接続されると共にメインプロセッサ１００に接続される。上述のメインプロセッサ１００、コプロセッサ２００、プログラムメモリ３００，４００、データメモリ５００、および外部バス６００は、同一プリント基板上に実装されてマイクロプロセッサシステムを構成する。もしくは、これらの一部あるいは全部を１つのシステムＬＳＩ上に集積してもよい。

【００２９】

ここで、メインプロセッサ１００は、プログラム中の第２の命令セットを解読してプログラムの進行を制御するための回路系として、プログラムメモリ３００をアクセスして第１の命令セットに属する命令（以下、単に第１の命令と称す）をフェッチする命令フェッチ回路１１０、フェッチされた第１の命令を一時的に保持する命令キュー１２０、第１の命令を解読する命令デコーダ１３０、メインプロセッサ側が現在処理中の命令のアドレス（即ちプログラムカウンタの値、以下ＰＣ値と称す）を更新する加算器１４０、ＰＣ値を保持するプログラムカウンタ１５０を有する。

【００３０】

また、このメインプロセッサ１００は、命令デコーダ１３０の解読結果に応じた演算を実行するための回路系として、この演算の入力データや演算結果（出力データ）を一時的に保持する汎用レジスタ１６０、この入力データに対して演算処理を施す算術論理ユニット（ＡＬＵ）１７０、この算術論理ユニット１７０により生成されたアドレスを使用して汎用レジスタ１６０と外部のデータバス６００との間でデータをロードまたはストアするデータアクセス回路１８０、および後述のコプロセッサ２００からの割り込み要求を受け付けるための割り込み要求受付回路１９０を有する。

さらに、この汎用のメインプロセッサ１００は、第１の命令セットをハードウェア上で実行することにより、第２の命令セットをソフトウェア上で間接的に実行可能なように構成されている。

【００３１】

一方、コプロセッサ２００は、第２の命令セットのうち、メインプロセッサ１００の管理下にあるデータを操作する必要を生ずる特定の命令に遭遇した場合、メインプロセッサ１００に通知して、この命令の実行を依頼するものである。

具体的には、コプロセッサ２００は、プログラム中の第２の命令セットを解読してプログラムの進行を制御するための回路系として、このコプロセッサ自体の特定の基本動作を制御するための制御データ（例えば動作開始や動作停止を指示するデータ）を一時的に保持するの制御レジスタ２１０、制御レジスタ２１０に保持された制御データに応じてプログラムメモリ４００から第２の命令セットに

属する命令（以下、単に第２の命令と称す）をフェッチするための命令フェッチ回路２２０、フェッチされた第２の命令を一時的に保持する命令キュー２３０、フェッチされた第２の命令を解読する命令デコーダ２４０、スタックされたデータの先頭アドレス（即ちスタックポインタの値、以下、ＳＰ値と称す）を命令デコーダの指示により更新するための加減算器２５０Ａ、ＳＰ値を保持するスタックポインタ２６０Ａ、現在処理中の命令のアドレス（即ちプログラムカウント値、以下ＰＣ値と称す）を命令デコーダの指示により更新するための加算器２５０Ｂ、ＰＣ値を保持するプログラムカウンタ２６０Ｂを有する。

なお、この発明において、「スタックされたデータの先頭アドレス」、すなわち「ＳＰ値」は、スタックされた最新データのアドレスを意味するものとする。

【００３２】

また、このコプロセッサ２００は、命令デコーダ２４０の解読結果に応じた演算を実行するための回路系として、第２の命令セットに属する命令を実行する過程で発生する各種のデータ（第２の命令の演算オペランド、ローカル変数、手続オペランド等）を保持するスタックメモリ２７０、このスタックメモリ上のオペランドに演算処理を施す算術論理ユニット（ＡＬＵ）２８０、および命令デコーダ２４０の解読結果に応じてメインプロセッサ１００に対し割り込み要求を発生する割り込み要求発生回路２９０を有する。なお、スタックメモリ２７０でのスタックは、アドレスの小さい方から大きい方に成長するものとし、スタックのデータ幅は例えば３２ビット（４バイト）である。

【００３３】

これら、制御レジスタ２１０、スタックポインタ２６０Ａ、プログラムカウンタ２６０Ｂ、およびスタックメモリ２７０は、コプロセッサ２００の内部バス２０１を介して外部のデータバス６００に接続されており、メインプロセッサ１００によりコプロセッサ側の制御レジスタ２１０やスタックポインタ２６０Ａの内容を直接的に設定可能なように構成されている。

【００３４】

ここで、図２を参照して、スタックメモリ２７０の動作概念を補足説明しておく。

このスタックメモリ２７０は、ＬＩＦＯ（後入れ先出し）方式のメモリであって、プログラムの実行中にデータを順々に積み重ねるようにして一時的に保持するものである。図２に示す例は、スタックポインタが示す先頭アドレスおよびその前のアドレスにそれぞれ収納された二つのデータに対して二項演算を施す場合におけるスタックメモリの内容の変化の様子を示す。

【００３５】

演算に先だって、同図（ａ）に示すように、アドレス「０ｘ１００４」～「０ｘ１０１４」には、データＤ１～Ｄ５が順次積み重ねられるように収納される。この場合、最後に収納されたデータＤ５のアドレス「０ｘ１０１４」が先頭アドレスとなり、スタックポインタの値とされる。この状態から、二項演算（例えば二項の加算演算）が実行されると、スタックポインタで示されるアドレス「０ｘ１０１４」のデータＤ５とアドレス「０ｘ１０１０」のデータＤ４とに対して演算が行われる。

【００３６】

具体的には、この演算の内容が加算演算であれば、データＤ４とデータＤ５とを加算して、その演算結果が算出される。この時点で、データＤ４，Ｄ５は不要となる。そこで、図２（ｂ）に示すように、演算結果として算出されたデータＤ６は、後続のデータＤ３の次のアドレス「０ｘ１０１０」に保存され、スタックポインタの値がアドレス「０ｘ１０１０」に更新される。

このように、スタックメモリには、演算を処理する過程で一時的にデータが保持され、スタックポインタによりその時々で有効なデータの先頭アドレスが指定されるようになっている。

【００３７】

図３に、命令デコーダ２４０の詳細な構成とその周辺を示す。

整列回路２２１は、命令フェッチ回路２２０の一部機能として構成されたもので、フェッチされた第２の命令から、オペレーションコードとオペランドコードとを抽出し、これらを整列して命令キュー２３０に出力する。

命令デコーダ２４０は、命令キュー２３０に保持されたオペレーションコードを解読するためのデコーダ２４０Ａ、オペレーションコードの解読結果に応じた

処理手順制御信号Ｓ２４０Ｂを発生する処理手順制御信号発生回路２４０Ｂ、ＳＰ値の増減値を発生するＳＰ用増減値発生回路２４０Ｃ、およびＰＣ値の増減値を発生するＰＣ用増減値発生回路２４０Ｄから構成される。

【００３８】

図４に、メインプロセッサ１００内の割り込み要求受付回路１９０の詳細と、コプロセッサ２００内の割り込み要求発生回路２９０の詳細を示す。

割り込み要求発生回路２９０は、割り込み処理用デコーダ２９１と、複数のフリップフロップ２９２とから構成される。割り込み処理用デコーダ２９１は、命令デコーダ２４０からの割り込み要求信号Ｓ２４１Ａに基づき、複数の専用割り込み要求信号Ｓ２９０Ａ～Ｓ２９０Ｆと、共用の割り込み要求信号Ｓ２９０Ｇとから構成される割り込みベクタＳ２９０を出力する。以下の説明において、適宜、複数の専用割り込み要求信号Ｓ２９０Ａ～Ｓ２９０Ｆを「専用割り込みベクタ」と称す。

【００３９】

ここで、専用割り込み要求信号Ｓ２９０Ａ～Ｓ２９０Ｆは、第２の命令セットのうち、コプロセッサが単独で処理不能な命令であって、比較的実行頻度の高い特定の１つ又は複数の命令に対して予め割り付けられている。つまり、コプロセッサがこの特定の命令に遭遇した場合、この遭遇した命令に割り付けられた専用割り込み要求信号が活性化されることにより、メインプロセッサは、この活性化された専用割り込み要求信号から、遭遇した命令を直接的に把握可能なようになっている。また、共用の割り込み要求信号Ｓ２９０Ｇは、コプロセッサが単独で処理不能な命令であって上述の特定の命令以外の命令に共通に割り付けられている。

このように、コプロセッサ２００は、第２の命令セットのうち、実行頻度の高い１つ又は複数の命令に対して予め割り付けられた１つ又は複数の専用割り込み要求信号を用いて「通知」を発行するように構成されている。

【００４０】

この実施の形態１では、コプロセッサ側で単独処理可能か否かの判断は、命令を処理（実行）する過程で、メインプロセッサ１００の管理下にあるデータ（例

えばデータメモリ５００に格納されたデータ）を操作する必要を生ずるか否かを基準として行われる。つまり、コプロセッサ２００は、メインプロセッサ側の管理下にあるデータを操作する必要を生じる命令については、単独処理不能と判断し、これ以外の命令については単独処理可能と判断する。コプロセッサ２００側で単独処理不能な特定の命令については、予め命令デコーダ２４０の内部に定義されている。

【００４１】

これら専用割り込み要求信号Ｓ２９０Ａ～Ｓ２９０Ｆおよび共用の割り込み要求信号Ｓ２９０Ｇは、複数のフリップフロップ（Ｆ／Ｆ）２９２に保持された後、割り込みベクタＳ２９０としてメインプロセッサ１００側に出力される。この図４に示す例では、命令デコーダ２４０からの３種類の割り込み要求信号Ｓ２４１Ａの論理値の組み合わせにより第２の命令が特定され、この命令の内容に応じて専用割り込み要求信号Ｓ２９０Ａ～Ｓ２９０Ｇの何れかが選択的に活性化されて出力される。

【００４２】

なお、専用割り込み要求信号Ｓ２９０Ａ～Ｓ２９０Ｆにプライオリティを設定してもよい。この場合、例えば、プライオリティの高い割り込み要求信号には、一つの命令が割り付けられ、プライオリティの低い割り込み要求信号には、複数の命令が割り付けられる。これにより、高いプライオリティを有する割り込み要求信号に割り付けられた命令については、即座に専用のハンドラが特定される。また、プライオリティの低い割り込み要求信号に割り付けられた命令については、後述するように、この割り込み要求信号に割り付けられた複数の命令で共有されるハンドラによって、実際にフェッチされた命令を特定するための処理が行われた後、この命令の割り込みを処理するためのルーチンが特定される。

【００４３】

一方、メインプロッセッサ側の割り込み要求受付回路１９０は、複数のフリップフロップ１９１と、割り込みハンドラアドレス発生回路１９２と、割り込みアドレスレジスタ１９３とから構成される。コプロセッサ側からの専用割り込み要求信号Ｓ２９０Ａ～Ｓ２９０Ｆおよび共用の割り込み要求信号Ｓ２９０Ｇは、複

数のフリップフロップ１９１にそれぞれ保持される。割り込みハンドラアドレス発生回路１９２は、コプロセッサ側からの専用割り込み要求信号Ｓ２９０Ａ～Ｓ２９０Ｇをエンコードして割り込みハンドラのアドレスを発生する。この割り込みハンドラのアドレスは、割り込みアドレスレジスタ１９３に保持された後に、上述の命令フェッチ回路１１０に出力される。

【００４４】

このように、割り込み要求受付回路１９０は、コプロセッサ２００からの専用割り込み要求信号Ｓ２９０Ａ～Ｓ２９０Ｆ（専用割り込みベクタ）をエンコードして、処理すべき特定の命令に対応する割り込みハンドラを特定するように構成される。

なお、メインプロセッサ１００は、コプロセッサ２００に対し、その動作を管理するための各種の操作を行う。この操作は、プログラムメモリ３００に格納されたプログラムをメインプロセッサ１００で実行することにより、ソフトウェア上で行われる。

【００４５】

以下、図５および図６に示すフローチャートに沿って、Ｊａｖａ言語などのスタック・アーキテクチャ命令で記述されたプログラムを実行する場合を例とし、この実施の形態１にかかるマイクロプロセッサシステムの動作を説明する。

ステップＳＴ１０：最初に、このマイクロプロセッサシステムのリセットが行われる。これにより、各部の動作が初期化され、このリセット後にメインプロセッサ１００が動作を開始し、コプロセッサ２００が停止状態に置かれる。

【００４６】

ステップＳＴ１１：続いて、メインプロセッサ１００は、ソフトウェア上での操作により、スタックの先頭アドレスをコプロセッサ２００内のスタックポインタ２６０Ａに設定し、コプロセッサ２００が実行すべきプログラムの先頭アドレスをプログラムカウンタ２６０Ｂに設定する。これにより、コプロセッサ２００のスタックポインタ２６０Ａおよびプログラムカウンタ２６０Ｂに初期値がそれぞれ設定される。

【００４７】

ステップＳＴ１２：続いて、メインプロセッサ１００は、ソフトウェア上での操作により、コプロセッサ２００内の制御レジスタ２１０に、「動作開始」を指示するための制御データ（例えばデータ「０」）を設定し、このコプロセッサ２００の動作を開始させる。

【００４８】

ステップＳＴ１３：続いて、動作を開始したコプロセッサ２００は、ハードウェア上での操作により、プログラムカウンタ２６０Ｂに設定されたアドレス（ＰＣ値）を参照して、第２の命令セットの中間コードが展開されたプログラムメモリ４００をアクセスし、このアドレスから命令フェッチ回路２２０により自発的に第２の命令をフェッチする。フェッチされた命令は命令キュー２３０に一旦保持される。

【００４９】

ステップＳＴ１４：続いて、コプロセッサ２００は、フェッチされた第２の命令を命令デコーダ２４０にて解読（デコード）する。

ステップＳＴ１５：続いて、コプロセッサ２００は、フェッチされた第２の命令がコプロセッサ側で単独処理可能か否かを判断する。

【００５０】

ここで、コプロセッサ２００が遭遇した第２の命令が、例えばＪａｖａ仮想マシンの「ｉａｄｄ」（整数のデータ同士を加算する命令）のように、メインプロセッサの管理下に有るデータを操作する必要のない命令、すなわちコプロセッサ側で処理可能な命令である場合（ステップＳＴ１５：ＹＥＳ）、コプロセッサ２００は、この命令を処理するための以下のステップＳＴ１６～ＳＴ１９を実行する。

【００５１】

ステップＳＴ１６：コプロセッサ２００は、ハードウェア上での操作により、スタックメモリ２７０から、演算処理の対象とされるデータを読み出す。上述の「ｉａｄｄ」命令の場合、ＳＰ値が表すアドレスと、このＳＰ値が表すアドレスから「４」を減算したアドレス（すなわち「ＳＰ値－４」）から、演算処理の対象とされるデータをそれぞれ読み出す。

【００５２】

ステップＳＴ１７：続いて、コプロセッサ２００は、ハードウェア上での操作により、スタックメモリ２７０から読み出したデータの演算を実行する。具体的には、上述の「ｉａｄｄ」命令の場合、ＳＰ値が表すアドレスから読み出したデータと、「ＳＰ値－４」が表すアドレスから読み出したデータとを加算する。

【００５３】

ステップＳＴ１８：続いて、コプロセッサ２００は、ハードウェア上での操作により、スタックメモリ２７０に演算結果を書き戻す。上述の「ｉａｄｄ」命令の場合、演算結果は、ＳＰ値が表すアドレスから「４」を減算したアドレス（すなわち「ＳＰ値－４」）に書き戻される。

【００５４】

ステップＳＴ１９：続いて、コプロセッサ２００は、ハードウェア上での操作により、現在処理の対象とされている命令の種類に応じて、スタックポインタ２６０Ａ内のＳＰ値と、プログラムカウンタ２６０Ｂ内のＰＣ値とを更新する。

【００５５】

具体的には、命令デコーダ２４０内のデコーダ２４０Ａの解読結果に応じて、ＳＰ用増減値発生回路２４０Ｃは、現在のＳＰ値に対する増減値を算出する。加減算器２５０Ａは、算出された増減値を現在のＳＰ値に加算してＳＰ値を更新し、これをスタックポインタ２６０Ａに設定する。また、命令デコーダ２４０内のデコーダ２４０Ａの解読結果に応じて、ＰＣ用増減値発生回路２４０Ｄは、現在のＰＣ値に対する増減値を算出する。加算器２５０Ｂは、算出された増減値を現在のＰＣ値に加算してＰＣ値を更新し、これをプログラムカウンタ２６０Ｂに設定する。

【００５６】

上述の「ｉａｄｄ」命令の場合、プログラムカウンタ２６０Ｂの現在のＰＣ値に「１」を加算し、スタックポインタ２６０Ａの現在のＳＰ値から「４」を減算して、ＰＣ値とＳＰ値を更新する。

この後、上述のステップＳＴ１３に戻り、次の第２の命令をフェッチして、この命令の内容に応じた処理が同様に行われる。

【００５７】

次に、フェッチされた第２の命令が、コプロセッサ２００側で処理不能な命令である場合の処理を説明する。

コプロセッサ２００は、遭遇した第２の命令が、例えばＪａｖａ仮想マシンの「ｉａｌｏａｄ」命令（整数の配列からデータをロードする命令）のように、メインプロセッサの管理下に有るデータを操作する必要のある命令、即ちコプロセッサが単独では処理不能な命令である場合、上述のステップＳＴ１５において否定的な判断を下す（ステップＳＴ１５：ＮＯ）。この場合、メインプロセッサ１００またはコプロセッサ２００において、この命令を処理するための以下のステップＳＴ２０～ＳＴ２６が実行される。

【００５８】

ステップＳＴ２０：コプロセッサ２００は、当該命令は専用のハンドラを持つか否か、すなわち当該命令に対して割り込み処理用のハンドラ（割り込みハンドラ）を特定するための専用割り込み要求信号が割り付けられているか否かを判断する。この実施の形態では、「ｉａｌｏａｄ」命令に対して、専用割り込み要求信号Ｓ２９０Ａが割り付けられており、「ｉａｌｏａｄ」命令は専用のハンドラを持つものとする。

【００５９】

ステップＳＴ２１：この場合（ステップＳＴ２０：ＹＥＳ）、コプロセッサ２００は、当該命令に割り付けられた専用割り込み要求信号Ｓ２９０Ａにより割り込みを発生させ、単独では処理不能な「ｉａｌｏａｄ」命令に遭遇したことをメインプロセッサ１００側に通知する。

ステップＳＴ２２：続いて、コプロセッサ２００は、ハードウェア上での操作により、制御レジスタ２１０に「動作停止」を指示するための制御データ（例えばデータ「１」）を設定し、動作を一旦停止する。

【００６０】

ステップＳＴ２３：続いて、専用割り込み要求信号Ｓ２９０Ａによりコプロセッサ２００から通知を受けたメインプロセッサ１００は、処理すべき命令に対応する割り込みハンドラの特定を行う。具体的には、割り込みハンドラアドレス発

生回路１９２によりコプロセッサ２００からの割り込み要求信号Ｓ２９０（Ｓ２９０Ａ～Ｓ２９０Ｇ）をエンコードして割り込みハンドラのアドレス（割り込みアドレス）を発生する。

【００６１】

ステップＳＴ２４：命令が特定されると、続いて、メインプロセッサ１００は、ソフトウェア上での操作により、スタックメモリ２７０またはデータメモリ５００から命令の実行（演算）に必要なデータを読み出して、命令の内容に応じた演算処理を実行する。例えば上述の「ｉａｌｏａｄ」命令の場合、ＳＰ値が表すアドレスで指定されるデータ（インデックス）と、このＳＰ値が表すアドレスから「4」を減算したアドレス（すなわち「ＳＰ値－４」）で指定されるデータ（配列リファレンス）を、スタックメモリ２７０からそれぞれ読み出す。そして、読み出された配列リファレンスで示される配列中のデータであって、インデックスに該当するデータをデータメモリ５００から読み出す。

【００６２】

ステップＳＴ２５：続いて、メインプロセッサ１００は、ソフトウェア上での操作により、コプロセッサ２００内のスタックメモリ２７０またはデータメモリ５００に演算結果を書き戻す。上述の「ｉａｌｏａｄ」命令の場合、メインプロセッサは、ＳＰ値から「4」を減算したアドレス（すなわち「ＳＰ値－４」）を指定して、スタックメモリ２７０に、データメモリ５００から読み出した上述のデータを書き込む。

【００６３】

ステップＳＴ２６：続いて、メインプロセッサ１００は、ソフトウェア上での操作により、コプロセッサ２００内の制御レジスタ２１０に、「動作開始」を指示するための制御データ（例えばデータ「0」）を設定し、コプロセッサ２００の動作を再開させる。

この後、上述のステップＳＴ１９に移行してＰＣ値とＳＰ値とを更新した後、ステップＳＴ１３に戻り、次の命令をフェッチして同様の処理が行われる。

【００６４】

次に、フェッチされた命令が、専用のハンドラを持たない場合の処理を説明す

る。

この場合、コプロセッサ２００は、上述のステップＳＴ２０において否定的判断を下す（ステップＳＴ２０：ＮＯ）。この場合、共用の割り込み要求信号Ｓ２９０Ｇにより通知が行われるが、複数の命令に共有される割り込み要求信号Ｓ２９０Ｇから、命令の種類を一義的に特定することはできない。そこでこの場合、メインプロセッサ１００またはコプロセッサ２００において、この命令を処理するための以下のステップＳＴ３０～ＳＴ３４，ＳＴ２４～ＳＴ２６が実行され、命令を特定するための処理が行われる。

【００６５】

ステップＳＴ３０：この場合、コプロセッサ２００は、ハードウェア上での操作により、共用の割り込み要求信号Ｓ２９０Ｇを活性化して、これをメインプロセッサ１００に出力する。

ステップＳＹ３１：続いて、コプロセッサ２００は、ハードウェア上での操作により、制御レジスタ２１０に、「動作停止」を指示するための制御データ（例えばデータ「1」）を設定し、その動作を停止する。

【００６６】

ステップＳＴ３２：続いて、共用割り込み要求信号Ｓ２９０Ｇによりコプロセッサ２００から通知を受けたメインプロセッサ１００は、共用の割り込みハンドラを特定し、割り込み処理を行う。すなわち、メインプロセッサ１００は、ソフトウェア上での操作により、プログラムカウンタ２６０ＢのＰＣ値を読み出し、現在処理中の命令のアドレスを把握する。

ステップＳＴ３３：続いて、メインプロセッサ１００は、ＰＣ値から把握されたアドレスを指定して、プログラムメモリ４００から命令のコードを読み出し、このコードから現在処理中の命令の種類を特定する。

【００６７】

ステップＳＴ３４：続いて、メインプロセッサ１００は、特定された命令の種類から、この命令に対応する共用の割り込みハンドラ中の処理ルーチンを特定する。

ステップＳＴ２４～ＳＴ２６：この後、メインプロセッサ１００は、上述と同

様に、特定された割り込みハンドラにより割り込み処理を行い、特定された命令に応じた演算を実行して結果を必要に応じてデータメモリ５００等に書き戻し、コプロセッサ２００の動作を再開させる。そして、上述のステップＳＴ１９に処理を移行させてＰＣ値およびＳＰ値を更新した後、処理を上述のステップＳＴ１３に戻し、次の命令に対する処理を同様に実行する。

【００６８】

以上のように、この実施の形態１では、コプロセッサ２００が、フェッチされた命令が単独で処理可能か否かを自ら判断する。そして、コプロセッサが単独では処理不能な命令に遭遇した場合、この命令の演算がメインプロセッサ１００のソフトウェア上で処理され、この命令の実行に伴って発生するＰＣ値およびＳＰ値の更新処理がコプロセッサ２００のハードウェア上で処理される。これにより、メインプロセッサ１００側では、メインプロセッサがコプロセッサから依頼された命令をソフトウェア上で処理する際、ＰＣ値やＳＰ値の更新処理を行う必要がなくなるので、インプロセッサ側のソフトウェアによる処理の負荷を軽減することができる。

【００６９】

また、この実施の形態では、コプロセッサ２００が、単独では処理不能な命令に遭遇した場合、この命令に専用に割り付けられた割り込みベクタを用いてメインプロセッサ１００に通知する。これにより、メインプロセッサ１００において、割り込みベクタから直接的に命令の種類を把握することができる。したがって、割り込みハンドラを特定する際、ソフトウェア上の処理の負荷を軽減することができる。

【００７０】

ここで、１命令あたりの動作クロック数が４０クロックと仮定すると、その内の１０数クロックが、命令の種類を特定するための処理に費やされ、命令を特定するための処理が、大きな負荷となる。したがって、上述のように、専用の割り込みベクタを設けることにより、割り込み処理の際に行われるインタプリタのデコードやプログラムカウンタの読み出しなど、命令の種類を特定するための処理が不要となり、メインプロセッサの処理上の負荷が大幅に軽減される。

【００７１】

上述の実施の形態１では、専用割り込み要求信号Ｓ２９０Ａ～Ｓ２９０Ｆが、コプロセッサ側で単独処理不能な命令であって比較的実行頻度の高い特定の複数の命令に対して予め割り付けられるものとしたが、これに限定されることなく、ひとつの専用割り込み要求信号を２以上の命令に割り付けるようにしてもよい。

【００７２】

ここで、例えば専用割り込み要求信号Ｓ２９０Ａを「ｉａｄｄ」命令および「ｉａｌｏａｄ」命令に割り付けた場合、この専用割り込み要求信号Ｓ２９０Ａから、コプロセッサが遭遇した命令を一義的に特定することはできない。そこで、この場合、共用の割り込み要求信号Ｓ２９０Ｇにより割り込み要求がなされた場合と同様に、コプロセッサ２００からプログラムカウンタのＰＣ値を読み取って命令の種類を特定する処理（ステップＳＴ３１～ＳＴ３４）を行えばよい。このようにひとつの専用割り込み要求信号を２以上の命令に割り付けると、ＰＣ値から命令を特定するための処理が必要となるが、より多くの種類の命令に対して専用割り込み要求信号を割り付けることができる。

【００７３】

また、この実施の形態１では、コプロセッサ２００が単独で処理できない命令として、メインプロセッサの管理下にあるデータを操作する必要が生じる命令を例としたが、これに限定されることなく、ソフトウェアで処理した方が効率的な命令に遭遇した場合に、メインプロセッサ側に処理を依頼するようにしてもよい。この種の命令の例として、上述のメインプロセッサの管理下にあるデータの操作を伴う命令の他に、スタックフレーム（プログラム上で定義された一群のスタックデータの集合）の入れ替えが発生する命令、浮動小数点演算を実行する命令などがある。

【００７４】

さらに、スタックアーキテクチャを採用するＲＩＳＣプロセッサで処理可能な命令をコプロセッサで実行し、それ以外の複雑な処理を必要とする命令をメインプロセッサで実行するように、メインプロセッサおよびコプロセッサがそれぞれ処理対象とする命令を区分けしてもよい。

【００７５】

図１に示した実施の形態１では、半導体集積回路のチップサイズ上の制限により、内蔵するスタックメモリ２７０の記憶容量が十分に確保できない場合がある。この場合、スタックメモリ２７０に書き込まれるデータのオーバーフロー及びアンダーフローを検出するための、ハードウェア手段もしくはソフトウェア手段を設ける必要がある。

【００７６】

ハードウェア手段としては、例えば、スタックポインタ２６０Ａが取り得る上限のアドレスと下限のアドレスとを値として保有するレジスタを設ければよい。このように構成すれば、スタックポインタの値が更新される度に、このレジスタの値と比較して、データのオーバーフロー及びアンダーフローを検出することができる。メインプロセッサ１００への検出の通知は、実行不能な命令に遭遇した場合と同様、割り込みを発生するか、あるいは、制御レジスタの特定のビットを変化させることによって実行することができる。

【００７７】

ソフトウェア手段としては、例えば、Ｊａｖａ言語では、バイトコードと呼ばれる中間コードを格納するファイル中に、プログラムの個々の実行単位毎に、スタック上で使用されるデータの量すなわちスタックフレームの大きさが予め明示されているので、各プログラムの実行前にメインプロセッサ上のソフトウェアによりこれをチェックするようにしておけばよい。

【００７８】

スタックのオーバーフローを検出した場合には、メインプロセッサ上のソフトウェアにより、スタックメモリ２７０中のデータを、データメモリ５００に待避させる。また、アンダーフローを検出した場合には、データメモリ５００に待避させたデータをスタックメモリ２７０に復元する。データの待避あるいは復元は、スタックメモリの全てのデータに対して一括して行うこともできるし、一部ずつ分割して行うこともできる。

【００７９】

この実施の形態１によれば、容量の制限された高速なスタックメモリ２７０を

コプロセッサの内部に集積したことによって、スタック上のデータをオペランドとする演算を算術論理ユニット２８０により高速に実行することが可能となる。さらに、データメモリ５００は、メインプロセッサ１００ともコプロセッサ２００とも異なる半導体集積回路上に実現すれば、オーバーフローの発生しない十分な記憶容量を確保することが容易に可能となる。

【００８０】

＜実施の形態２＞

次に、この発明の実施の形態２を説明する。

Ｊａｖａ言語の命令体系では、スタックデータの先頭（上位）の４つのデータをレジスタ内に持つことにより、命令を効率的に実行し得るものとなっている。逆に言えば、先頭の４つのデータ以外のスタックデータについては、あえてレジスタに保持する必要はなく、外部のメモリに保持させておいても、命令の実行には支障がない。このように、スタックデータを外部のメモリに保持させることにより、スタックデータの容量を飛躍的に拡大することができる。

この実施の形態２では、上述のようにスタックデータの先頭の４つのデータのみをレジスタ内に保持し、Ｊａｖａ言語の処理に適したマイクロプロセッサシステムについて説明する。

【００８１】

図７に、この実施の形態２に係るプロセッサシステムの構成を示す。このプロセッサシステムは、前述の図１に示す実施の形態１にかかるプロセッサシステムの構成において、スタックメモリ２７０に代えてスタックトップレジスタ２７０Ａとスタックメモリ２７０Ｂとを有するコプロセッサ２００Ａを備える。また、この実施の形態では、命令キュー２３０は内部バス２０１にも接続され、メインプロセッサ１００側から、命令キュー２３０の内容を直接的に読み出すことが可能なようになっている。

【００８２】

ここで、スタックトップレジスタ２７０Ａは、算術論理ユニット２８０に接続され、スタックデータのうち、先頭側の４つのデータ（所定数のデータ）を保持するものである。このスタックトップレジスタ２７０Ａは、リードとライトが同

時に且つ高速に行い得るように構成されたものである。また、スタックメモリ２７０Ｂは、上述の実施の形態１にかかるスタックメモリ２７０に相当するものであって、外部バス６００とスタックトップレジスタ２７０Ａとに接続されて、コプロセッサ２００Ａの外部に設けられ、スタックデータの上位の４つのデータの下位側のデータを保持する。この構成の場合、スタックポインタ２６０ＡのＳＰ値は、スタックメモリ２７０Ｂの先頭アドレスを示す。

【００８３】

ここで、図８を参照して、スタックトップレジスタとスタックメモリの一般的な動作概念を補足説明しておく。スタックトップレジスタおよびスタックメモリは、上述のスタックメモリ２７０と同様にＬＩＦＯ方式でデータの書き込みと読み出しが行われるものであって、同図（ａ）に示す例では、スタックデータＤ１１～Ｄ１５のうち、先頭の２つのデータＤ１４，Ｄ１５がスタックトップレジスタに保持される。

【００８４】

以下、２項演算を行う場合を例として、スタックトップレジスタおよびスタックメモリの内容の変化の様子を示す。演算に先だって、図８（ａ）に示すように、アドレス「０ｘ１００４」～「０ｘ１０１４」には、データＤ１１～Ｄ１５が順次積み重ねられるように収納される。この場合、スタックメモリに最後に収納されたデータＤ１３のアドレス「０ｘ１００Ｃ」が先頭アドレスとなり、スタックポインタの値とされる。この状態から、二項演算（例えば二項の加算演算）が実行されると、スタックトップレジスタ内のデータＤ１４とデータＤ１５とに対して演算がなされる。

【００８５】

具体的には、この演算が「加算」であれば、データＤ１４とデータＤ１５とが加算され、その演算結果としてデータＤ１６が得られる。この時点で、データＤ１４，Ｄ１５は不要となる。そこで、図８（ｂ）に示すように、演算結果として算出されたデータＤ１６は、空いたアドレスの下位側、すなわち後続のデータＤ１３の次のアドレス「０ｘ１０１４」に保存される。このとき先頭側のデータＤ１６およびその後続のデータＤ１３の２つのデータは、スタックトップレジスタ

側に保持される。このように、スタックトップレジスタには、演算を処理する過程で、スタックデータの先頭側の所定数のデータが一時的に保持される。

【００８６】

なお、図８による補足説明では、スタックトップレジスタは、先頭側の２つのデータを保持するものとしたが、図７に示すこの実施の形態２に係る構成では、スタックデータのうち、先頭の４つのデータをスタックトップレジスタ２７０Ａに保持するものである。ただし、この発明は、これに限定されるものではなく、スタックトップレジスタのデータ数は、必要に応じた任意の数に設定される。

【００８７】

以下、図９に示すフローチャートに沿って、スタックフレーム（プログラム上で定義された一群のスタックデータの集合）を入れ替える場合を含む動作を例として、この実施の形態２の動作を説明する。

最初に、このマイクロプロセッサシステムがリセットされる。これにより、各部の動作が初期化され、このリセット後にメインプロセッサ１００が、動作を開始し、コプロセッサ２００Ａが停止状態に置かれる。

【００８８】

ステップＳＴ１０１：この状態から、メインプロセッサ１００は、ソフトウェア上での操作により、スタックの先頭アドレスをコプロセッサ２００Ａ内のスタックポインタ２６０Ａに設定し、コプロセッサ２００Ａが実行すべきプログラムの先頭アドレスをプログラムカウンタ２６０Ｂに設定する。これにより、コプロセッサ２００Ａのスタックポインタ２６０Ａおよびプログラムカウンタ２６０Ｂに初期値がそれぞれ設定される。

【００８９】

ステップＳＴ１０２：続いて、メインプロセッサ１００は、ソフトウェア上での操作により、コプロセッサ２００Ａ内の制御レジスタ２１０に、「動作開始」を指示するための制御データ（例えばデータ「０」）を設定し、コプロセッサ２００Ａの動作を開始させる。

【００９０】

ステップＳＴ１０３：続いて、動作を開始したコプロセッサ２００Ａは、ハー

ドウェア上での操作により、プログラムカウンタ２６０Ｂに設定されたアドレスを参照して、第２の命令セットの中間コードが展開されたプログラムメモリ４００をアクセスし、このアドレスから自発的に第２の命令をフェッチする。フェッチされた命令は命令キュー２３０に保持される。そして、コプロセッサ２００Ａは、フェッチされた命令を命令デコーダ２４０にて解読（デコード）する。

ステップＳＴ１０４：続いて、コプロセッサ２００Ａは、フェッチされた命令がスタックフレームを変更する命令か否かを判断する。

【００９１】

ここで、スタックトップレジスタが先頭側の２つのデータを保持するものである場合を例とし、図１０を参照して、スタックフレームの変更の概念を補足説明しておく。この例では、フレーム変更前において、スタックメモリ２７０Ｂには、引数データＤ２１、ローカル変数Ｄ２２、コンテキストＤ２３、演算データＤ２４、および引数データＤ２５の一部（引数データｄ２５－ｎ）が保持されている。一方、スタックトップレジスタ２７０Ａには、引数データＤ２５の残り（引数データｄ２５－１，ｄ２５－２）が保持されている。これら先頭側のデータＤ２１～Ｄ２５が、１つのスタックフレームを構成する。

【００９２】

フレーム変更を行うと、変更前のフレームに新たなデータが積み重ねられ、この先頭側のデータ群が新たなスタックフレームを構成する。この図に示す例では、それまでのデータＤ２５に積み重ねるようにして、ローカル変数Ｄ２６、コンテキストＤ２７が保持され、その先頭側の引数データＤ２５～コンテキストＤ２７が新たなスタックフレームを構成する。

【００９３】

フレーム変更後では、スタックメモリ２７０Ｂには、引数データＤ２１～引数データＤ２５に加え、新たなデータとして、ローカル変数Ｄ２６、およびコンテキストＤ２７の一部（コンテキストｄ２７－ｎ）が保持される。一方、スタックトップレジスタ２７０Ａには、コンテキストＤ２７の残り（コンテキストｄ２７－１，ｄ２７－２）が保持される。この一連のスタックフレームの変更処理は、メインプロセッサ１００のソフトウェア上で行われる。

【００９４】

上述のように、スタックフレームを変更する場合、スタックポインタの値が大きく変わる。この処理を行う場合、スタックトップレジスタ２７０Ａとスタックメモリ２７０Ｂとにわたってスタックデータを操作する必要が生じ、メインプロセッサ１００での処理が複雑化する。

そこで、以下のステップで説明するように、スタックフレームの変更処理を実行する前に、スタックトップレジスタ２７０Ａの全ての内容を、予めスタックメモリ２７０Ｂに移しておく。これにより、スタックフレームを変更するための処理では、スタックメモリ２７０Ｂのみを参照すればよく、この処理の負荷を軽減することができる。

【００９５】

ここで、再び説明を図９に示すフローチャトに戻す。

ステップＳＴ１０５：上述のステップＳＴ１０４において、フェッチされた命令がスタックフレームを変更する命令であると判断された場合（ステップＳＴ１０４：ＹＥＳ）、コプロセッサ２００Ａは、スタックトップレジスタ２７０Ａに保持されている全ての内容をスタックメモリ２７０Ｂに書き込む。そして、コプロセッサ２００Ａは、この書き込みによるスタックメモリ２７０Ａのデータの増分に応じて、スタックポインタ２６０ＡのＳＰ値を更新する。

【００９６】

ステップＳＴ１０６：続いて、コプロセッサ２００Ａは、フェッチされた命令が専用の割り込みハンドラを持つか否かを判断して、割り込み要求を発生し、メインプロセッサ側において割り込みハンドラの特定が行われる。このステップＳＴ１０６での処理は、上述の実施の形態１にかかるステップＳＴ２０～ＳＴ２３，ＳＴ３０～ＳＴ３４（図６参照）による処理と同様に行われる。

【００９７】

ステップＳＴ１０７：続いて、スタックフレームを変更するための演算処理を行う。具体的には、メインプロセッサ１００は、全てのスタックデータが保持されたスタックメモリ２７０Ｂまたはデータメモリ５００から処理に必要なデータを読み出し、新たなスタックフレームを準備する。続いて、コプロセッサ２００

Ａ内のスタックポインタ２６０ＡのＳＰ値（アドレス）およびプログラムカウンタ２６０ＢのＰＣ値（アドレス）を読み出して保存する。続いて、このスタックポインタ２６０Ａおよびプログラムカウンタ２６０Ｂに新たな値（アドレス）を書き込む。そして、スタックメモリ２７０Ｂまたはデータメモリ５００に演算結果を書き戻す。続いて、コプロセッサ２００Ａの制御レジスタ２１０に「操作開始」指示する制御データを設定し、コプロセッサ２００Ａに動作を再開させる。

【００９８】

ステップＳＴ１０８：動作を再開したコプロセッサ２００Ａは、メインプロセッサ１００の演算結果が収納されたスタックメモリ２７０Ｂから、スタックトップレジスタ２７０Ａのデータ数分のデータを読み込み、これをスタックトップレジスタ２７０Ａに保持する。そして、この読み込みによるスタックメモリ２７０Ｂのデータの減少分に応じて、スタックポインタ２６０ＡのＳＰ値を更新する。

ステップＳＴ１０９：この後、コプロセッサ２００Ａは、プログラムカウンタ２６０ＢのＰＣ値を、「０」を加算したものとして更新して、処理を上述のステップＳＴ１０３に戻し、次の命令を処理対象とする。

【００９９】

ステップＳＴ１１０：次に、上述のステップＳＴ１０４において、スタックフレームを変更する命令でない場合（ステップＳＴ１０４：ＮＯ）、コプロセッサ２００Ａは、フェッチされた命令が、スタックを増加させる命令か否かを判断する。つまり、スタックメモリにスタックされるデータが発生する場合には、スタックメモリに空き領域を確保する必要が生じる。

【０１００】

このステップでの判断は、換言すれば、演算に先だって、この空き領域を予め確保するか否かを判断するものである。なお、スタックが減少する場合には、演算を実行する前に、スタックポインタを減少させることは許されない。なぜなら、スタックポインタを減少させることにより、演算に必要なデータが無効とされる場合が発生し得るからである。

【０１０１】

ステップＳＴ１１１：ここで、スタックを増加させる命令である場合（ステッ

プＳＴ１１０：ＹＥＳ）、コプロセッサ２００Ａは、必要分のスタックトップレジスタ２７０Ａの内容をスタックメモリ２７０Ｂに書き込み、スタックポインタ２６０Ａの値を更新する。また、スタックを増加させる命令でない場合（ステップＳＴ１１０：ＮＯ）、このステップＳＴ１１１をパスする。

【０１０２】

ステップＳＴ１１２：続いて、コプロセッサ２００Ａは、フェッチされた命令が単独で処理可能な命令か否かを判断し、この判断結果に応じて、コプロセッサ２００Ａのハードウェア上での処理、またはメインプロセッサ１００のソフトウェア上での処理が行われ、この命令に対する処理が適応的に行われる。このステップＳＴ１１２での処理は、具体的には、上述の実施の形態１にかかるステップＳＴ１５～１８，ＳＴ２０～ＳＴ２６，ＳＴ３０～ＳＴ３４（図５参照）による処理と同様である。

【０１０３】　この実施の形態２では、メインプロセッサ１００は、コプロセッサ２００Ａの命令キュー２３０を直接的にアクセスできるので、メインプロセッサ１００がコプロセッサから依頼された命令を処理する場合、コプロセッサ２００Ａの命令キュー２３０を参照して、処理すべき命令の種類を特定し、この命令に対応する割り込みハンドラを特定することができる。これにより、専用の割り込みベクタが割り付けられていない命令であっても、即座に命令の種類が特定される。

【０１０４】

ステップＳＴ１１３：続いて、コプロセッサ２００Ａは、フェッチされた命令がスタックを減少させる命令であるか否かを判断する。

ステップＳＴ１１５：ここで、スタックを減少させる命令である場合（ステップＳｔ１１３：ＹＥＳ）、コプロセッサ２００Ａは、必要分のスタックトップレジスタの内容をスタックメモリから読み込み、スタックポインタ２６０Ａの値を更新する。また、スタックを減少させる命令でない場合（ステップＳＴ１１３：ＮＯ）、このステップＳＴ１１４をパスする。

この後、処理は上述のステップＳＴ１０９に移行し、プログラムカウンタ２６０ＢのＰＣ値が更新される。そして、上述のステップＳＴ１０３に戻され、次の

命令が処理の対象とされる。

【０１０５】

以上のように、この実施の形態２によれば、コプロセッサ２００Ａにおいて、命令デコーダにより解読された命令がスタックポインタを非連続的な領域に変更する場合（例えば上述のスタックフレームを変更するような場合）、メインプロセッサ１００に通知する前に、スタックトップレジスタ２７０Ａの全てのスタックデータをスタックメモリに書き出し、メインプロセッサ１００での処理が終了して動作が復帰した後に、スタックメモリ２７０Ｂからスタックトップレジスタ２７０Ａに先頭側のデータを読み出す。

【０１０６】

これにより、メインプロセッサ１００は、割り込みハンドラ内にて、スタックデータを単一のメモリ空間上で取り扱うことができ、スタックトップレジスタ２７０Ａのデータを扱う場合の処理が簡略化される。したがって、スタックトップレジスタ２７０Ａを有するアーキテクチャと親和性が高いＪａｖａなどのプログラム言語の命令を効率的に実行することが可能となる。

【０１０７】

また、この実施の形態２によれば、コプロセッサ２００Ａは、スタックメモリ２７０Ｂを外部に持つので、スタックメモリ２７０Ｂの記憶容量を大きくすることができる。可能であれば、このスタックメモリ２７０Ｂをコプロセッサ２００Ａの内部に持つようにしてもよい。

【０１０８】

＜実施の形態３＞

次に、この発明の実施の形態３を説明する。

図１１に、この実施の形態３にかかるマイクロプロセッサシステムの構成を示す。

このマイクロプロセッサシステムは、上述の図７に示す実施の形態２に係る構成において、スタックトップレジスタ２７０Ａとスタックメモリ２７０Ｂとの間に、キャッシュメモリ２７０Ｃを有するコプロセッサ２００Ｂを備える。このキャッシュメモリ２７０Ｃは、スタックメモリ２７０Ｂに保持されたデータの一部

をキャッシュするものである。

なお、この実施の形態３では、スタックメモリ２７０Ｂと外部バス６００との間の接続は切り離されている。

【０１０９】

このように、キャッシュメモリ２７０Ｃを備えることにより、見かけ上、外部のスタックメモリ２７０Ｂを高速にアクセスすることが可能となり、処理の高速化が図れる。この場合、スタックメモリ２７０Ｂとキャッシュメモリ２７０Ｃとの間のデータの整合を保つ必要上、スタックメモリ２７０Ｂは、必ずキャッシュメモリ２７０Ｃを介してアクセスされる。このため、スタックメモリ２７０Ｂは外部バス６００からの直接アクセスが不能なように、この外部バス６００との接続が切り離されている。

すなわち、メインプロセッサ１００からスタックメモリ２７０Ｂに格納されるべきデータにアクセスするためには、外部バス６００とコプロセッサ２００の内部バス２０１とを経由し、さらにキャッシュメモリ２７０Ｃを通してアクセスしなければならない。

【０１１０】

この実施の形態３によれば、キャッシュメモリ２７０Ｃを介してスタックメモリ２７０Ｂがアクセスされるので、スタックメモリ２７０Ｂを高速にアクセスすることができ、仮にスタックメモリ２７０Ｂが高速にアクセスできないものであっても、見かけ上の処理を高速化することができる。

また、この実施の形態３によれば、コプロセッサ２００Ｂは、スタックメモリ２７０Ｂを外部に持つので、スタックメモリ２７０Ｂの記憶容量を大きくすることができる。

【０１１１】

さらに、この実施の形態３によれば、キャッシュメモリ２７０Ｃ自体にキャッシュの対象となるデータを管理する機能が付加されているので、外部に接続されたスタックメモリ２７０Ｂの記憶容量を十分大きくしてオーバーフローに対する対策を講じる必要がない。すなわち、実施の形態１では、ハードウェア手段またはソフトウェア手段により、オーバーフローあるいはアンダーフローの検出を行

っているが、この実施の形態３では、オーバーフロー検出手段、アンダーフロー検出手段が不要であり、また、オーバーフローまたはアンダーフローの検出に伴うメモリ入れ換え作業が不要となるので、チップサイズの縮小および制御用ソフトウェアのプログラムコードサイズの縮小が実現可能となる。

【０１１２】

＜実施の形態４＞

次に、この発明の実施の形態４を説明する。

図１２に、この実施の形態４にかかるマイクロプロセッサシステムの構成を示す。同図に示すように、このプロセッサシステムは、１つのメインプロセッサ１００に対して、複数のコプロセッサ２００Ｘ，２００Ｙ，２００Ｚが設けられている。コプロセッサ２００Ｘ，２００Ｙ，２００Ｚは、前述の実施の形態１にかかるスタックメモリ２７０に相当する専用のスタックメモリ２７０ＸＭ，２７０ＹＭ，２７０ＺＭをそれぞれ持つ。

【０１１３】

これらコプロセッサ２００Ｘ，２００Ｙ，２００Ｚは、外部バス６００を介してメインプロセッサ１００に接続されると共に、外部バス６００Ｂを介してプログラムメモリ４００に接続される。また、プログラムメモリ４００およびデータメモリ５００は外部バス６００を介してメインプロセッサ１００に接続される。各コプロセッサからの割り込み要求Ｓ２９０Ｈはメインプロセッサ１００に直接入力される。

【０１１４】

この実施の形態４では、各コプロセッサのスタックメモリ２７０ＸＭ，２７０ＹＭ，２７０ＺＭには、それぞれのコプロセッサに固有のローカルなデータ（例えばローカル変数、ローカルな演算データ、ローカルなワーキングデータ）が格納される。このローカルなデータとしては、例えばＪａｖａ言語の場合、プログラムに記述されたタスクの単位である「スレッド」に相当するデータの集合が格納される。

【０１１５】

この実施の形態４では、メインプロセッサ１００が処理すべき命令とコプロセ

ッサ２００Ｘ，２００Ｙ，２００Ｚが処理すべき命令は、データメモリ５００上に設けられたヒープ領域（オブジェクトが書き込まれる領域）を操作する必要があるか否かにより区別される。すなわち、ヒープ領域を操作する必要のある命令については、メインプロセッサ１００で処理を行い、ヒープ領域を操作する必要のない命令については、コプロセッサ２００Ｘ，２００Ｙ，２００Ｚで処理を行う。

【０１１６】

また、メインプロセッサ１００は、データメモリ５００上のヒープ領域に作成されたオブジェクトへのアクセスが複数のコプロセッサのスレッドで発生したときに、各スレッド間の排他制御を行う。

具体的には、各コプロセッサは、固有のローカルデータが格納されたスタックメモリ２７０ＸＭ，２７０ＹＭ，２７０ＺＭをアクセスして実行可能な命令のみを処理の対象とする。

【０１１７】

また、メインプロセッサ１００は、各コプロセッサ間の調整を伴う命令を処理の対象とする。例えば、Ｊａｖａ言語の「ｉｎｖｏｋｅｖｉｒｔｕａｌ」命令は、ヒープ領域中のオブジェクトに対し複数の処理が記述されたプログラム中の特定の処理を選択的に実行する。このとき、実行の対象となるオブジェクトが他のスレッドによって排他的に使用されている場合には、その使用が終了するまでプログラムの実行を待たなければならない。メインプロセッサ１００は、実行中のスレッドの終了を待ち、次の要求のあったコプロセッサにアクセスして、プログラムカウンタに実行すべき処理が記述されたアドレスを設定後、該当するコプロセッサの制御レジスタを動作開始に設定する。

【０１１８】

ただし、Ｊａｖａ言語の「ガベージコレクション」すなわち、ヒープ領域上のオブジェクトを操作して、不要となったデータを廃棄する処理については、コプロセッサによる割り込みとは無関係に、メインプロセッサ１００が実行する。

なお、メインプロセッサ１００は、「ガベージコレクション」の処理中に各コプロセッサからの割り込みを受け付けないように、予めこの割り込みをマスクす

ることにより、ガベージコレクションに伴う格納場所の移動中のオブジェクトの書き換えを防止する。

【０１１９】

この実施の形態４によれば、複数のコプロセッサ２００Ｘ，２００Ｙ，２００Ｚにより、マルチタスクの処理が可能となり、Ｊａｖａ言語などのようにマルチスレッドの概念を有するプログラムを高速に実行することが可能となる。

また、各コプロセッサでの処理と並行して、メインプロセッサ１００がガベージコレクションなどの処理を行うことができ、したがって全体の処理時間を有効に短縮することができる。

【０１２０】

さらに、メインプロセッサ１００は、各コプロセッサからの割り込み処理をマスクすることにより、オブジェクト操作要求を容易に制御することができ、このため、ガベージコレクションを容易に且つリアルタイムに行うことができる。

さらにまた、データメモリ５００上のヒープ領域に作成されたオブジェクトへのアクセスが複数のコプロセッサのスレッドで発生したときに、各スレッド間の排他制御をメインプロセッサ上で一括処理するので、コプロセッサ側に排他制御のための手段やプログラムを用意しなくてもよく、また、メインプロセッサ上の排他制御処理プログラムが簡略化できる。

【０１２１】

以上、この発明の実施の形態１ないし４を説明したが、この発明は、これらの実施の形態に限られるものではなく、この発明の要旨を逸脱しない範囲の設計変更等があっても本発明に含まれる。例えば、上述の各実施の形態では、コプロセッサ側からメインプロセッサ側に割り込みを発生させることにより、通知するものとしたが、これに限定されることなく、たとえばコプロセッサ側に状態レジスタを設け、この状態レジスタに割り込み要求を通知するためのデータを書き込んでおき、この状態レジスタをメインプロセッサが定期的にアクセスすること（ポーリング）により、コプロセッサが処理不能な特定の命令に遭遇したことをメインプロセッサ側で把握するものとしてもよい。これにより、結果的に、メインプロセッサ側に「通知」が行われ、メインプロセッサでその処理が実行される。

【０１２２】

なお、この発明は、見方を変えれば、コプロセッサが遭遇した命令の内容に応じて、命令の処理形態を選択するものとして把握することができる。例えば、コプロセッサが持つスタックデータのみで処理が可能な命令についてはコプロセッサ自身で処理を行い、またデータメモリ上のヒープ領域を操作する必要のある命令についてはメインプロセッサで処理を行い、さらにスタックフレームを変更する必要のある命令についてもメインプロセッサで処理を行うように、処理の主体を選択する。

【０１２３】

【発明の効果】

以上説明したように、この発明によれば、プログラムに記述された命令を実行するマイクロプロセッサシステムであって、第１の命令セットをハードウェア上で実行すると共に第２の命令セットをソフトウェア上で実行するメインプロセッサと、前記メインプロセッサの管理下で動作して前記第２の命令セットをハードウェア上で実行するコプロセッサとを備えたので、回路規模の増大を抑えつつ、特定の命令セット（第２の命令セット）を高速に実行することが可能となる。

【０１２４】

また、前記コプロセッサが、前記第２の命令セットのうち、前記メインプロセッサの管理下にあるデータを操作する必要を生ずる特定の命令に遭遇した場合、前記メインプロセッサに通知を発行して当該命令の実行を依頼するようにしたので、メインプロセッサのソフトウエアが有効に活用されると共にコプロセッサのハードウェアの規模を必要最小限に抑えることが可能となり、したがって全体としてハードウェアの規模を有効に抑えることが可能となる。

【０１２５】

さらに、前記コプロセッサが、前記第２の命令セットのうち、実行頻度の高い所定の複数の命令に対して予め割り付けられた専用割り込みベクタを用いて前記通知を発行するようにしたので、メインプロセッサにおいて、コプロセッサが遭遇した命令が即座に特定される。したがって、メインプロセッサ側において、コプロセッサからの依頼により処理すべき命令を識別するための処理を省くことが

可能となる。

【０１２６】

さらにまた、前記コプロセッサが、前記第２の命令セットに属する命令を実行する過程で発生するデータを保持するためのスタックメモリと、該スタックメモリ内の最新データのアドレスを保持するスタックポインタとを有し、前記特定の命令を実行する過程で発生する処理のうち、前記スタックメモリのスタックポインタを更新するための処理を実行するハードウエア資源を備えたので、メインプロセッサが依頼を受けてこの命令を処理する過程において、スタックポインタを更新するための処理を省くことができる。したがって、メインプロセッサのソフトウエア上での処理が簡略化されると共に、コプロセッサのハードウェアが有効に活用され、メインプロセッサでのソフトウェアによる処理時間を短縮することができる。

【０１２７】

さらにまた、前記コプロセッサが、現在処理中の前記第２の命令セットに属する命令のアドレスを保持するためのプログラムカウンタを有し、前記特定の命令を実行する過程で発生する処理のうち、前記プログラムカウンタを更新するための処理を実行するハードウエア資源を備えたので、メインプロセッサに処理を依頼する場合であっても、プログラムカウンタを更新するための処理については、コプロセッサのハードウェア上で行われ、メインプロセッサがプログラムカウンタを更新するためのソフトウェア上の処理を省くことができる。したがって、メインプロセッサのソフトウエア上の処理が簡略化されると共に、コプロセッサのハードウェアが有効に活用され、メインプロセッサでのソフトウェアによる処理時間を有効に短縮することができる。

【０１２８】

さらにまた、前記コプロセッサが、前記通知を行う必要が発生したことを表す情報を保持するための状態レジスタを有し、前記メインプロセッサが、前記状態レジスタを定期的にアクセスして、該状態レジスタの内容から前記コプロセッサが前記特定の命令に遭遇したことを把握して、前記特定の命令を実行するようにしたので、メインプロセッサにより、コプロセッサが特定の命令に遭遇したか否

かが定期的に把握され、メインプロセッサに対して割り込みベクタを用いて割り込み処理を行う必要がなくなる。したがって、コプロセッサからメインプロセッサへの通知を簡略化することが可能となる。

【０１２９】

さらにまた、前記メインプロセッサが、前記コプロセッサからの前記専用割り込みベクタをエンコードして、処理すべき前記特定の命令に対応する割り込みハンドラを特定する割り込み要求受付回路を備えたので、メインプロセッサにおいて、コプロセッサから入力する専用割り込みベクタに基づき直接的に割り込みハンドラが特定される。したがって、メインプロセッサ側で起動すべき割り込みハンドラを特定するためのソフトウエア上の処理が不要となり、メインプロセッサでの処理時間を短縮することが可能となる。

【０１３０】

さらにまた、前記メインプロセッサが、前記コプロセッサの命令キューを参照して、処理すべき前記特定の命令に対応する割り込みハンドラを特定するようにしたので、メインプロセッサにおいて、コプロセッサの命令キューを参照してコプロセッサが遭遇した命令が把握され、直接的に処理ルーチンが特定される。したがって、メインプロセッサ側で起動すべき割り込みハンドラを特定するためのソフトウエア上の処理が簡略化され、メインプロセッサでの処理時間を短縮することが可能となる。

【０１３１】

さらにまた、前記コプロセッサが、スタックアーキテクチャを有するようにしたので、コプロセッサにおいて、サブルーチンの頻繁な使用を伴う命令や、ゼロオペランドのアドレッシングを伴う命令の処理が効率的に行われる。したがってＪａｖａなどのスタックアーキテクチャに基づくインタプリタ言語の命令を効率的に実行することが可能となる。

【０１３２】

さらにまた、スタックアーキテクチャを採用する前記コプロセッサが、スタックデータのうち、先頭側の所定数のデータを保持するスタックトップレジスタと、外部に設けられたスタックメモリと前記スタックトップレジスタとの間に設け

られ、前記スタックメモリに保持されたデータの一部をキャッシュするキャッシュメモリと、を含むようにしたので、スタックデータのうち、先頭側の所定数のデータをアクセスする頻度の高いプログラム言語の命令を効率的に実行することができると共に、スタックメモリの容量を適応的に拡大することが可能となる。

【０１３３】

さらにまた、プログラムに記述された複数の処理内容に対応づけて、複数の前記コプロセッサを備えたので、プログラム中の複数の処理を並列処理することが可能となり、処理を高速化することが可能となる。

【図面の簡単な説明】

【図１】　この発明の実施の形態１にかかるマイクロプロセッサシステムの構成を示すブロック図である。

【図２】　この発明の実施の形態１にかかるスタックメモリの動作概念を説明するための説明図である。

【図３】　この発明の実施の形態１にかかる命令デコーダの詳細とその周辺の構成を示すブロック図である。

【図４】　この発明の実施の形態１にかかる割り込み要求発生回路と割り込み要求受付回路の詳細な構成を示すブロック図である。

【図５】　この発明の実施の形態１にかかるマイクロプロセッサシステムの動作（命令がコプロセッサで処理可能な場合）の流れを示すフローチャートである。

【図６】　この発明の実施の形態１にかかるマイクロプロセッサシステムの動作（命令がコプロセッサで処理不能な場合）の流れを示すフローチャートである。

【図７】　この発明の実施の形態２にかかるマイクロプロセッサシステムの構成を示すブロック図である。

【図８】　この発明の実施の形態２にかかるスタックトップレジスタの動作概念を説明するための説明図である。

【図９】　この発明の実施の形態２にかかるマイクロプロセッサシステムの動作の流れを示すフローチャートである。

【図１０】　この発明の実施の形態２にかかるスタックトップレジスタの動作概念（スタックフレームを変更する場合）を説明するための説明図である。

【図１１】　この発明の実施の形態３にかかるマイクロプロセッサシステムの構成を示すブロック図である。

【図１２】　この発明の実施の形態４にかかるマイクロプロセッサシステムの構成を示すブロック図である。

【符号の説明】

１００：メインプロセッサ

１１０：命令フェッチ回路

１２０：命令キュー

１３０：命令デコーダ

１４０：加算器

１５０：プログラムカウンタ

１６０：汎用レジスタ

１７０：算術論理ユニット

１８０：データアクセス回路

１９０：割り込み要求受付回路

１９１：フリップフロップ

１９２：割り込みハンドラアドレス発生回路

１９３：割り込みアドレスレジスタ

２００：コプロセッサ

２００Ｘ～２００Ｚ：コプロセッサ

２０１：内部バス

２１０：制御レジスタ

２２０：命令フェッチ回路

２２１：整列回路

２３０：命令キュー

２４０：命令デコーダ

２４０Ａ：デコーダ

２４０Ｂ：処理手順制御信号発生回路

２４０Ｃ：ＳＰ用増減値発生回路

２４０Ｄ：ＰＣ用増減値発生回路

２５０Ａ：加減算器

２５０Ｂ：加算器

２６０Ａ：スタックポインタ

２６０Ｂ：プログラムカウンタ

２７０：スタックメモリ

２７０Ａ：スタックトップレジスタ

２７０Ｂ：スタックメモリ

２７０Ｃ：キャッシュメモリ

２８０：算術論理ユニット

２９０：割り込み要求発生回路

２９１：割り込み処理用デコーダ

２９２：フリップフロップ

３００：プログラムメモリ（メインプロセッサ用）

４００：プログラムメモリ（コプロセッサ用）

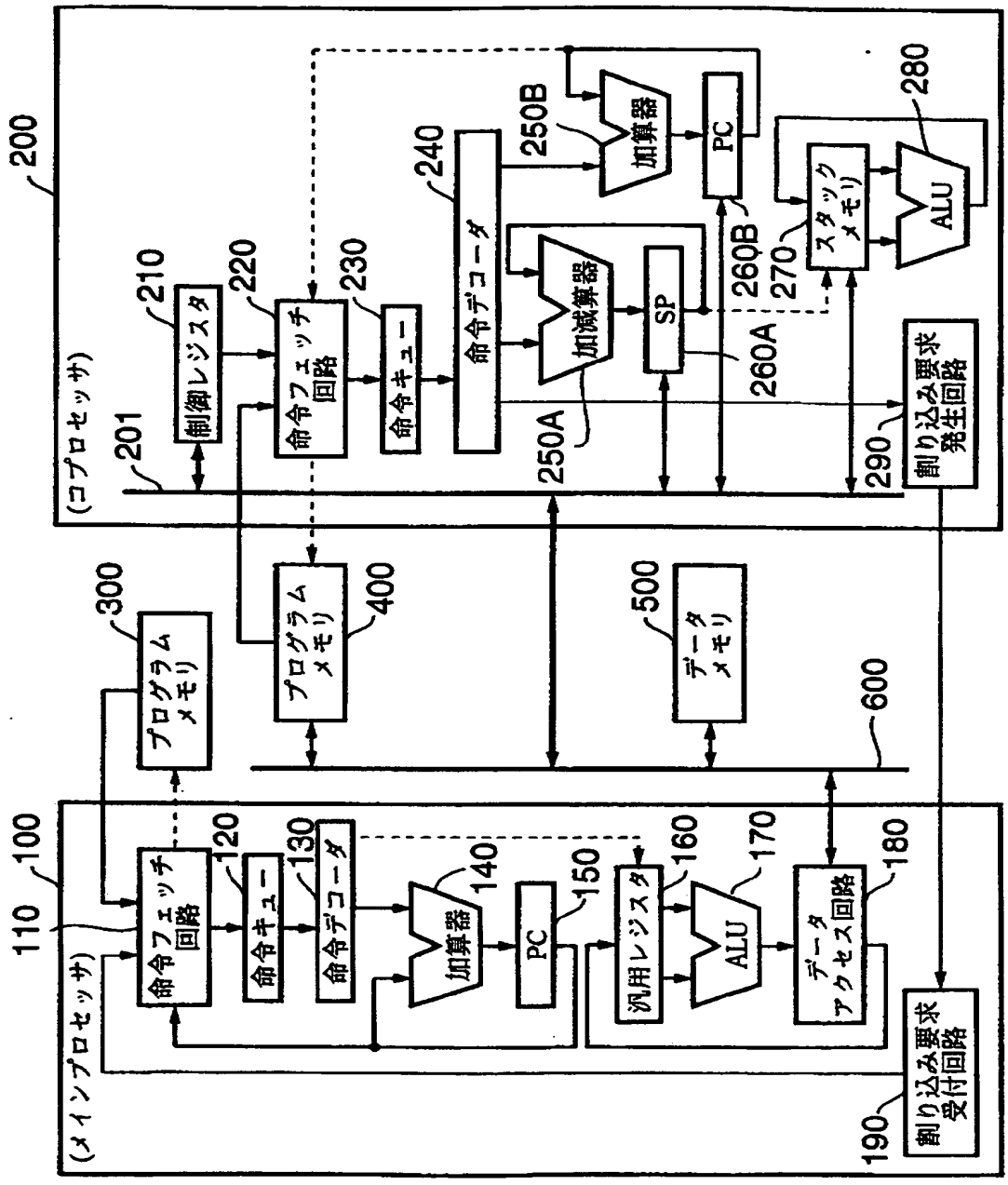
５００：データメモリ

６００，６００Ｂ：外部バス

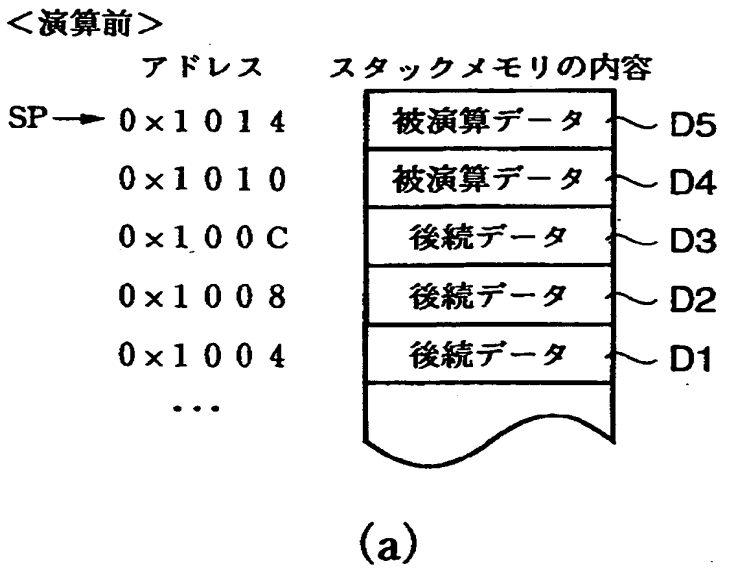
ＳＴ１０～ＳＴ２６，ＳＴ３０～ＳＴ３４，ＳＴ１０１～ＳＴ１１４：ステップ

【書類名】　　図面

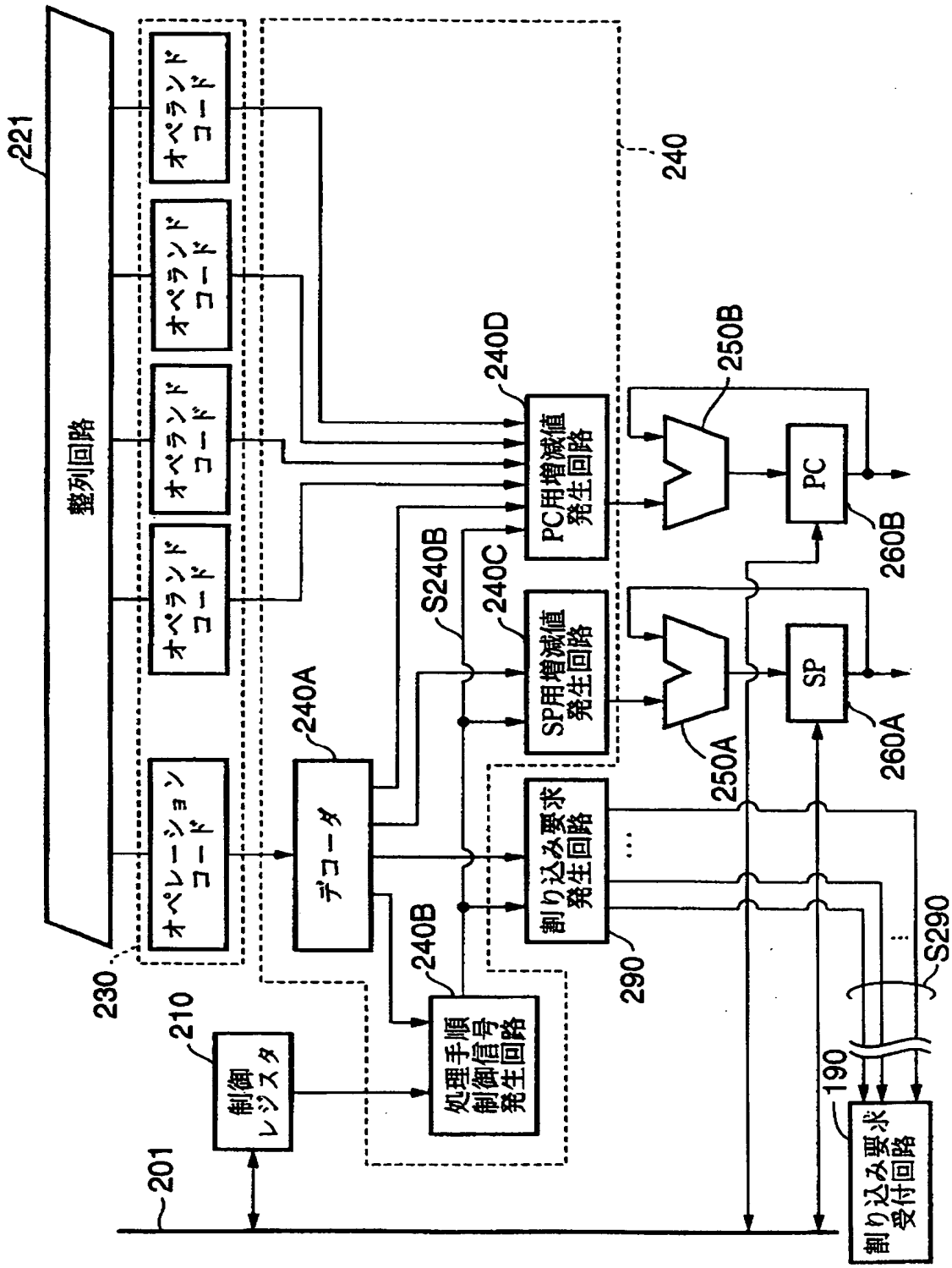
【図１】


200
(コプロセッサ)
210
制御レジスタ
201
220
命令フェッチ回路
230
命令キュー
240
命令デコーダ
250B
加算器
250A
加減算器
PC
SP
260B
260A
270
スタックメモリ
ALU
280
290
割り込み要求発生回路
300
プログラムメモリ
400
プログラムメモリ
500
データメモリ
600
100
110
(メインプロセッサ)
命令フェッチ回路
120
命令キュー
130
命令デコーダ
140
加算器
PC
150
汎用レジスタ
160
ALU
170
データアクセス回路
180
190
割り込み要求受付回路

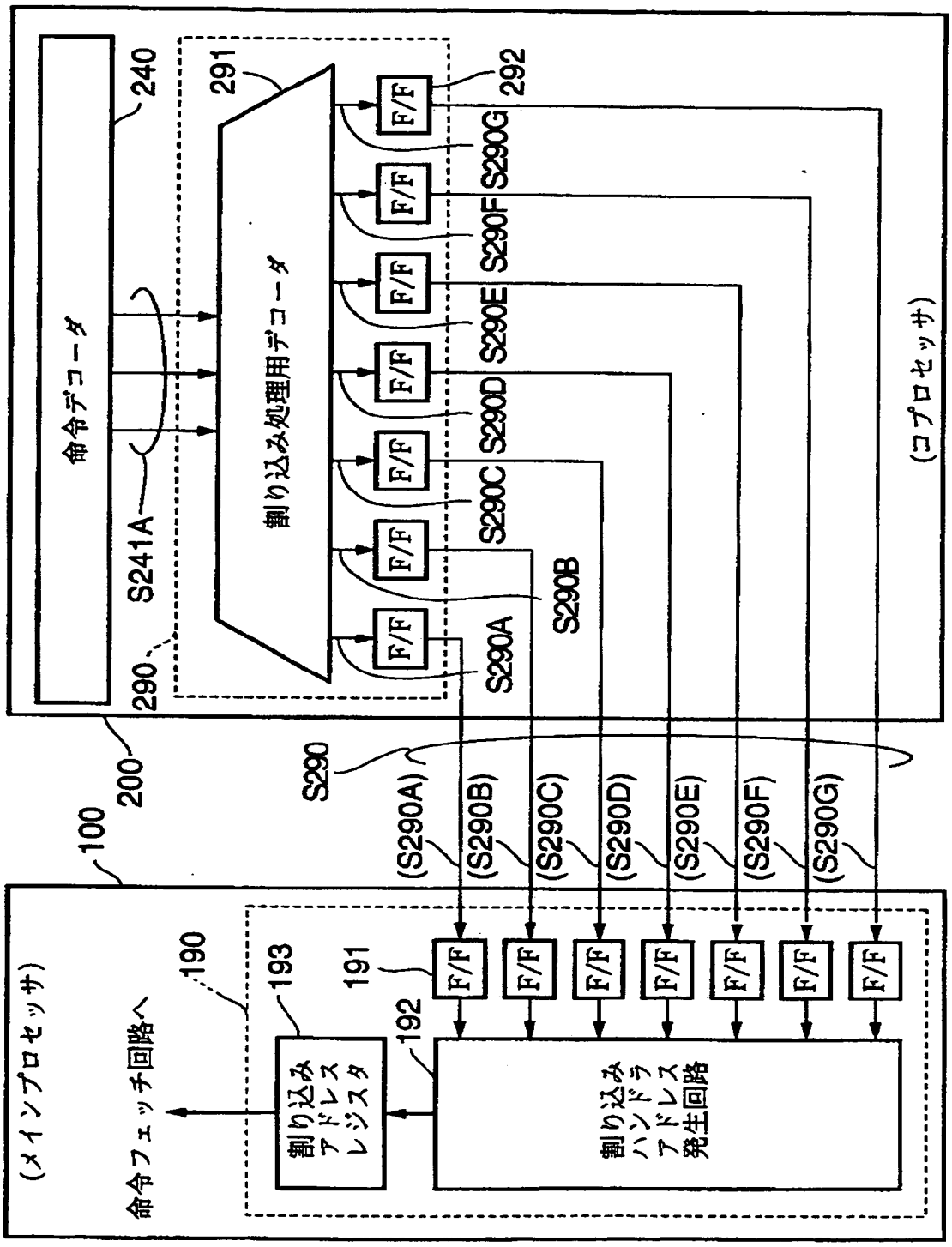
【図２】


<演算前>
アドレス
スタックメモリの内容
SP
0×1014
0×1010
0×100C
0×1008
0×1004
被演算データ
被演算データ
後続データ
後続データ
後続データ
D5
D4
D3
D2
D1
(a)
<演算後>
アドレス
スタックメモリの内容
0×1014
SP
0×1010
0×100C
0×1008
0×1004
結果データ
後続データ
後続データ
後続データ
D6
D3
D2
D1
(b)

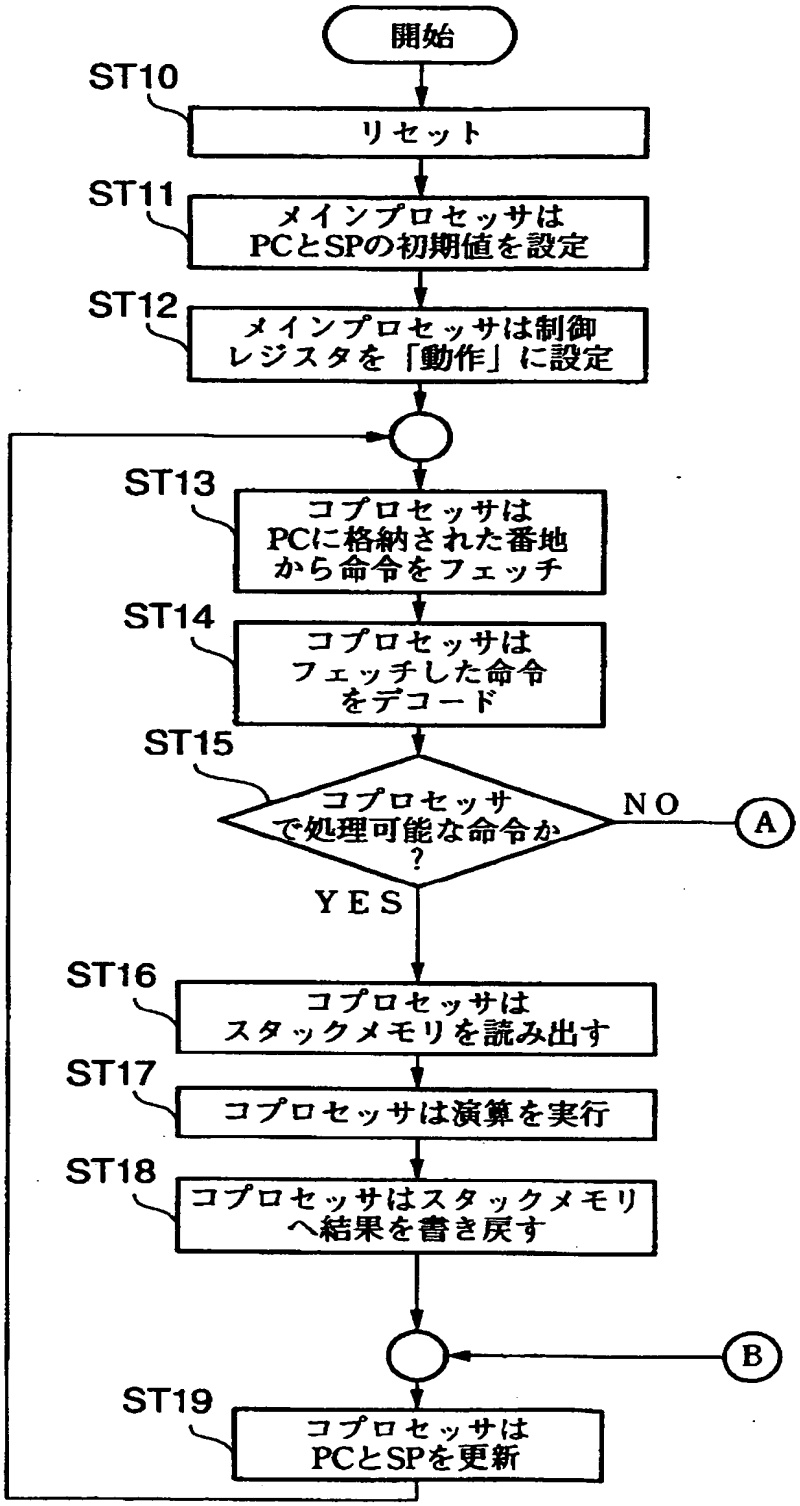
【図３】


221
整列回路
オペランドコード
オペランドコード
オペランドコード
オペランドコード
オペレーションコード
230
240
240D
PC用増減値発生回路
250B
PC
260B
S240B
240C
SP用増減値発生回路
250A
SP
260A
240A
デコーダ
割り込み要求発生回路
290
S290
240B
処理手順制御信号発生回路
210
制御レジスタ
201
190
割り込み要求受付回路

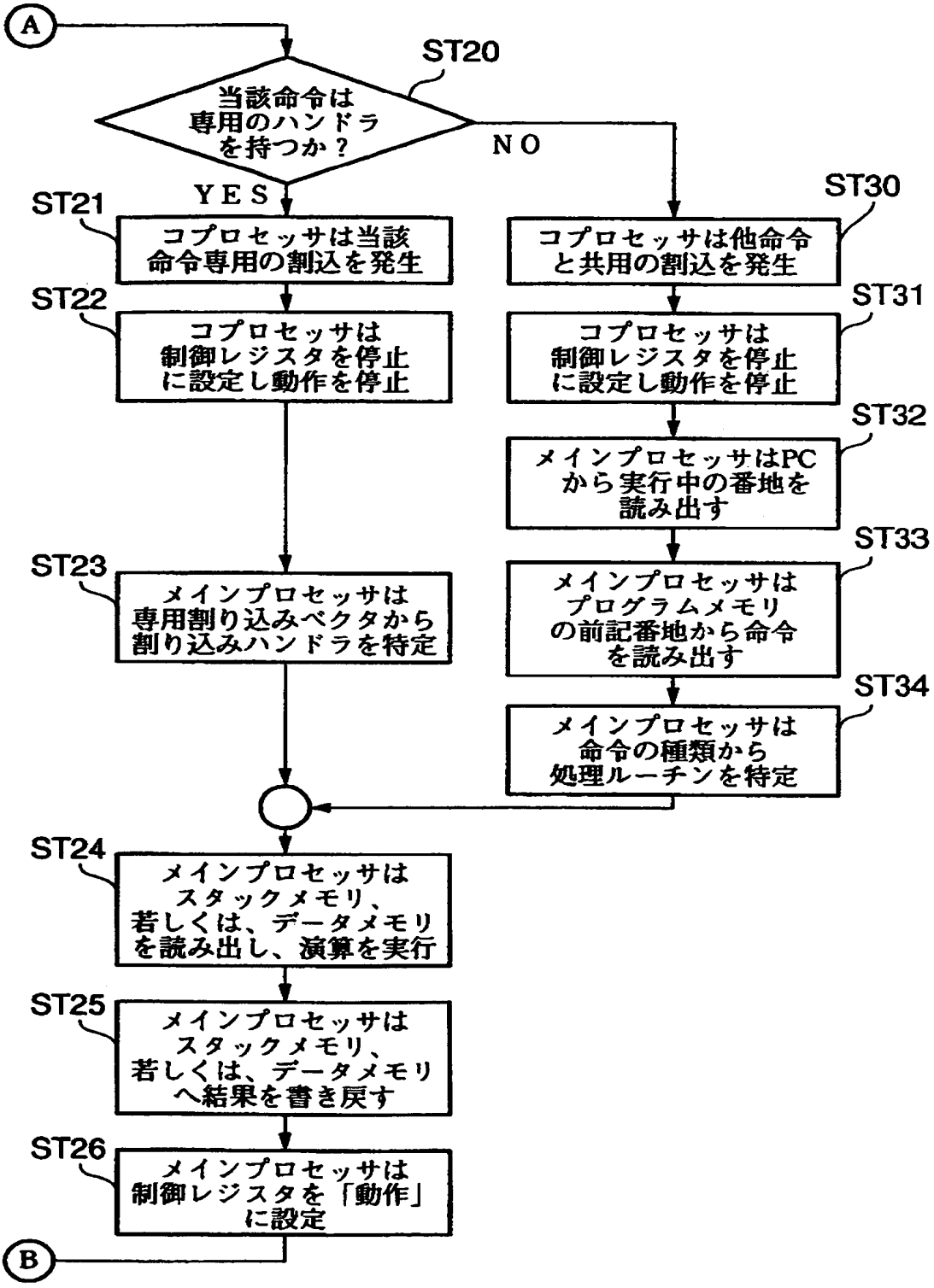
【図４】


(メインプロセッサ)
100
命令フェッチ回路へ
190
193
191
192
割り込み
アドレス
レジスタ
割り込み
ハンドラ
アドレス
発生回路
F/F
(S290A)
(S290B)
(S290C)
(S290D)
(S290E)
(S290F)
(S290G)
200
S290
(コプロセッサ)
290
命令デコーダ
240
S241A
割り込み処理用デコーダ
291
292
S290A
S290B
S290C
S290D
S290E
S290F
S290G

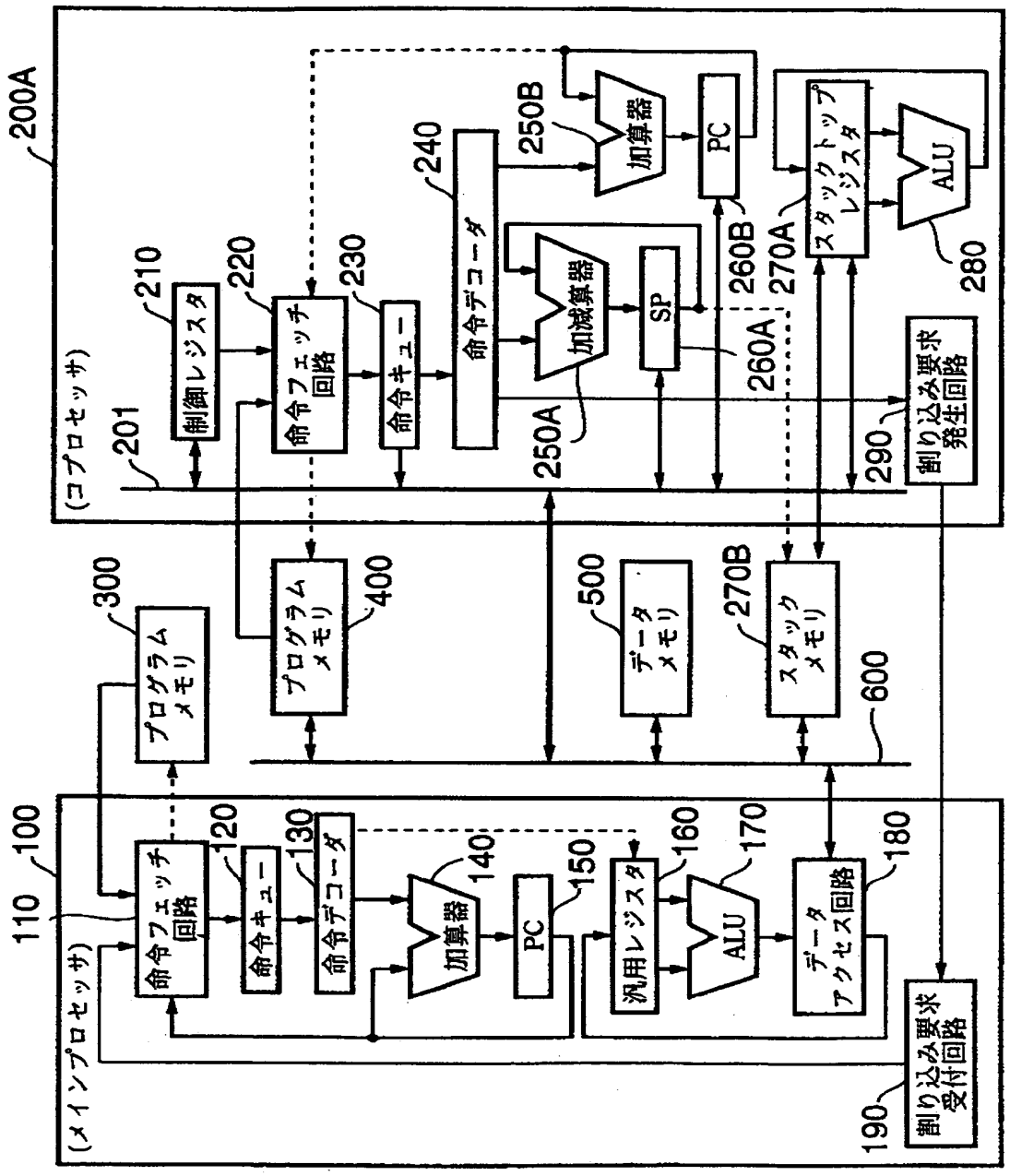
【図５】


開始
ST10
リセット
ST11
メインプロセッサは
PCとSPの初期値を設定
ST12
メインプロセッサは制御
レジスタを「動作」に設定
ST13
コプロセッサは
PCに格納された番地
から命令をフェッチ
ST14
コプロセッサは
フェッチした命令
をデコード
ST15
コプロセッサ
で処理可能な命令か
？
NO
A
YES
ST16
コプロセッサは
スタックメモリを読み出す
ST17
コプロセッサは演算を実行
ST18
コプロセッサはスタックメモリ
へ結果を書き戻す
B
ST19
コプロセッサは
PCとSPを更新

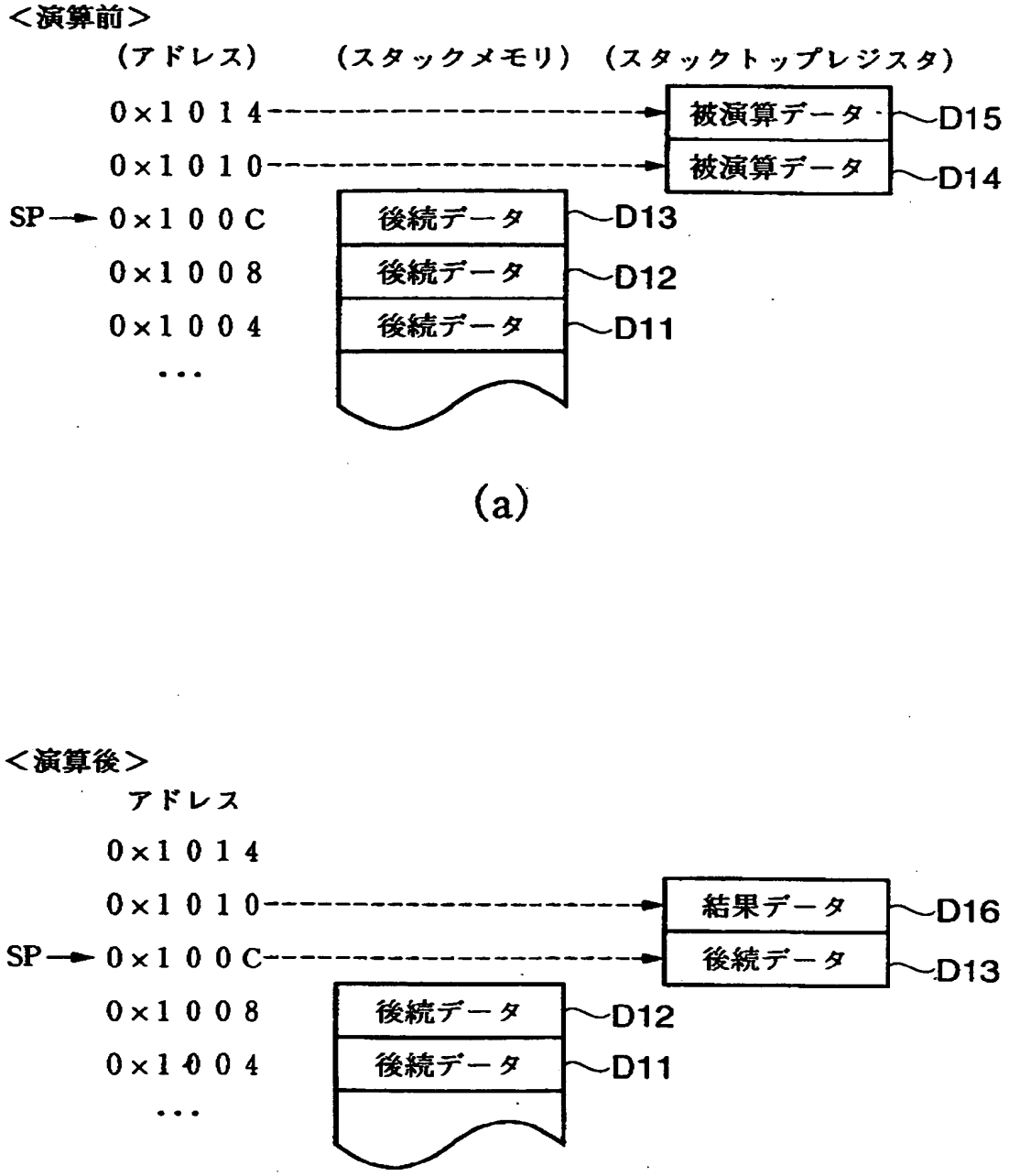
【図６】


A
ST20
当該命令は
専用のハンドラ
を持つか？
NO
YES
ST21
コプロセッサは当該
命令専用の割込を発生
ST22
コプロセッサは
制御レジスタを停止
に設定し動作を停止
ST23
メインプロセッサは
専用割り込みベクタから
割り込みハンドラを特定
ST30
コプロセッサは他命令
と共用の割込を発生
ST31
コプロセッサは
制御レジスタを停止
に設定し動作を停止
ST32
メインプロセッサはPC
から実行中の番地を
読み出す
ST33
メインプロセッサは
プログラムメモリ
の前記番地から命令
を読み出す
ST34
メインプロセッサは
命令の種類から
処理ルーチンを特定
ST24
メインプロセッサは
スタックメモリ、
若しくは、データメモリ
を読み出し、演算を実行
ST25
メインプロセッサは
スタックメモリ、
若しくは、データメモリ
へ結果を書き戻す
ST26
メインプロセッサは
制御レジスタを「動作」
に設定
B

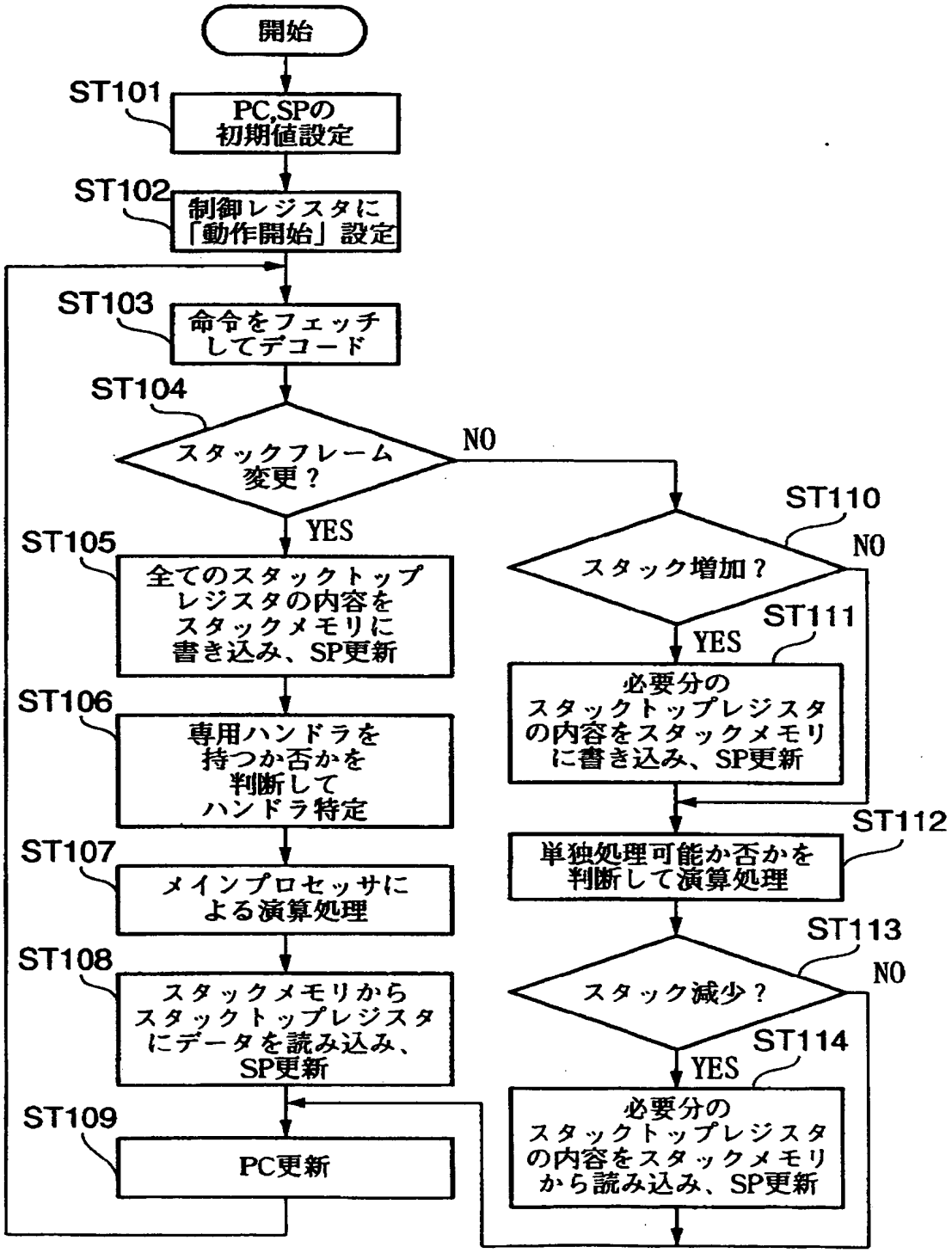
【図７】


200A
(コプロセッサ)
201
制御レジスタ
210
命令フェッチ回路
220
命令キュー
230
命令デコーダ
240
加減算器
250A
加算器
250B
SP
260A
PC
260B
スタックトップレジスタ
270A
ALU
280
割り込み要求発生回路
290
プログラムメモリ
300
プログラムメモリ
400
データメモリ
500
スタックメモリ
270B
600
100
(メインプロセッサ)
110
命令フェッチ回路
120
命令キュー
130
命令デコーダ
140
加算器
150
PC
160
汎用レジスタ
170
ALU
180
データアクセス回路
190
割り込み要求受付回路

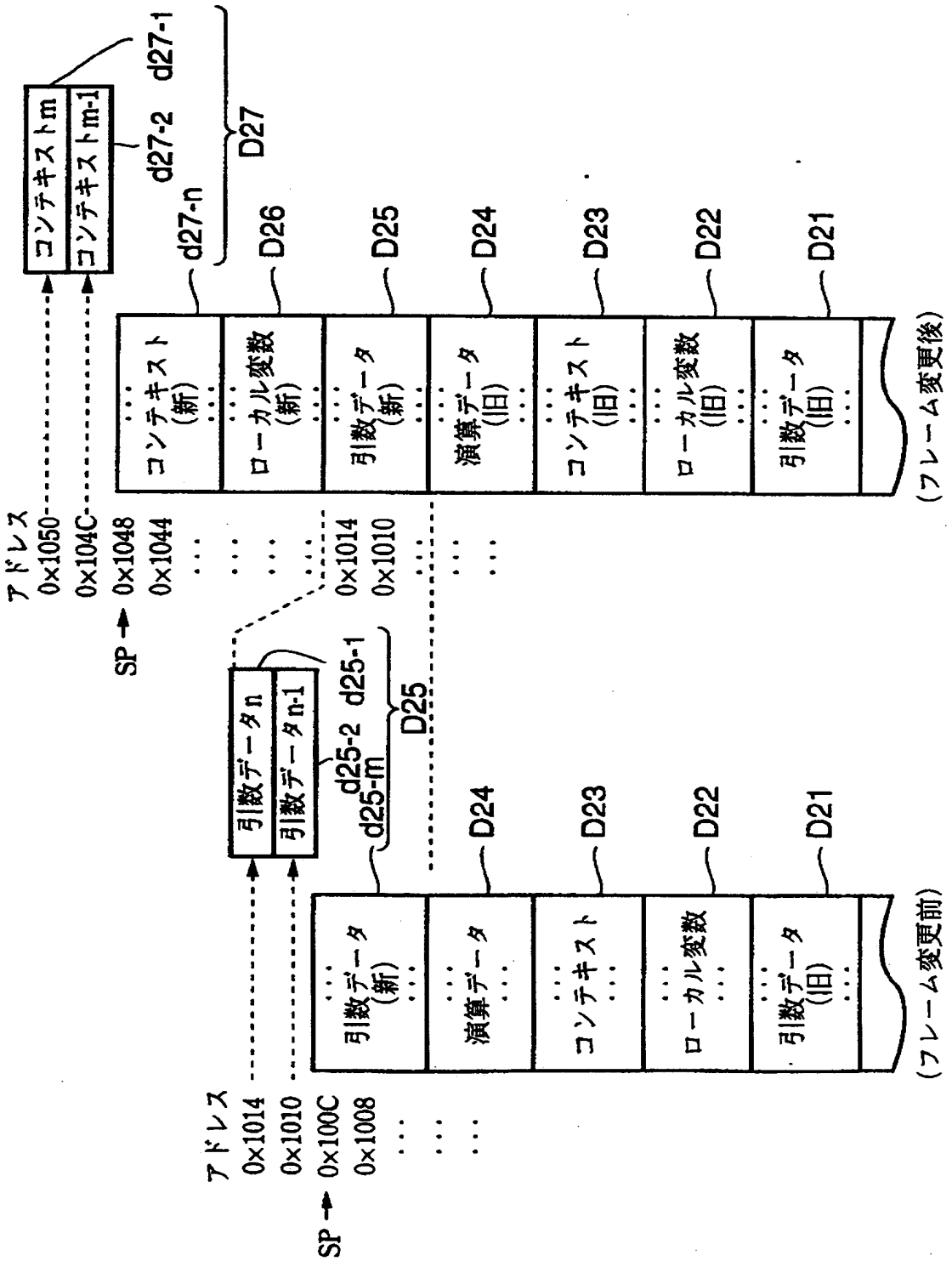
【図８】


<演算前>
（アドレス）
（スタックメモリ）
（スタックトップレジスタ）
0×1014
0×1010
SP
0×100C
0×1008
0×1004
被演算データ
被演算データ
後続データ
後続データ
後続データ
D15
D14
D13
D12
D11
(a)
<演算後>
アドレス
0×1014
0×1010
SP
0×100C
0×1008
0×1004
結果データ
後続データ
後続データ
後続データ
D16
D13
D12
D11
(b)

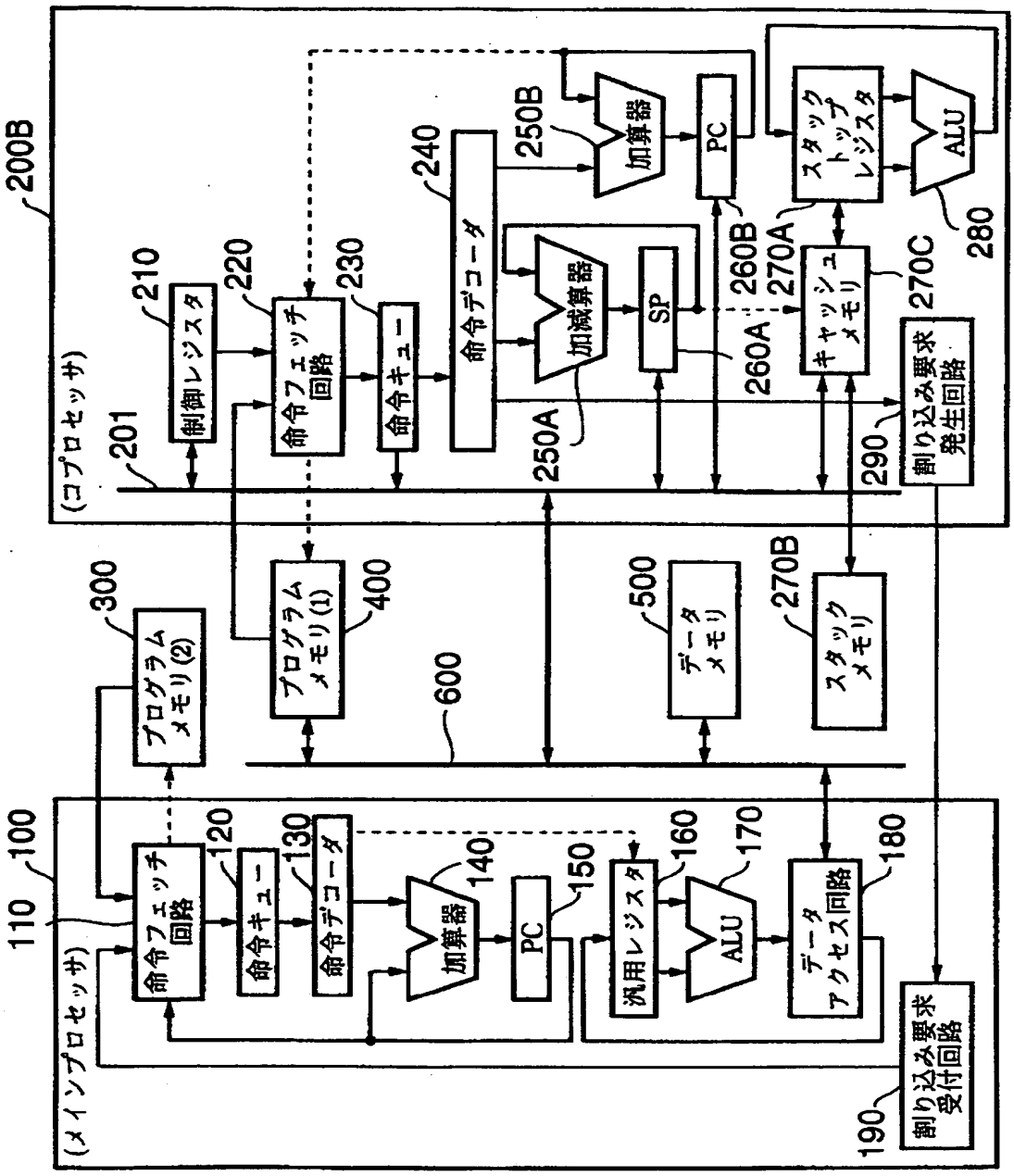
【図９】


開始
ST101
PC,SPの
初期値設定
ST102
制御レジスタに
「動作開始」設定
ST103
命令をフェッチ
してデコード
ST104
スタックフレーム
変更？
NO
YES
ST105
全てのスタックトップ
レジスタの内容を
スタックメモリに
書き込み、SP更新
ST106
専用ハンドラを
持つか否かを
判断して
ハンドラ特定
ST107
メインプロセッサに
よる演算処理
ST108
スタックメモリから
スタックトップレジスタ
にデータを読み込み、
SP更新
ST109
PC更新
ST110
スタック増加？
NO
YES
ST111
必要分の
スタックトップレジスタ
の内容をスタックメモリ
に書き込み、SP更新
ST112
単独処理可能か否かを
判断して演算処理
ST113
スタック減少？
NO
YES
ST114
必要分の
スタックトップレジスタ
の内容をスタックメモリ
から読み込み、SP更新

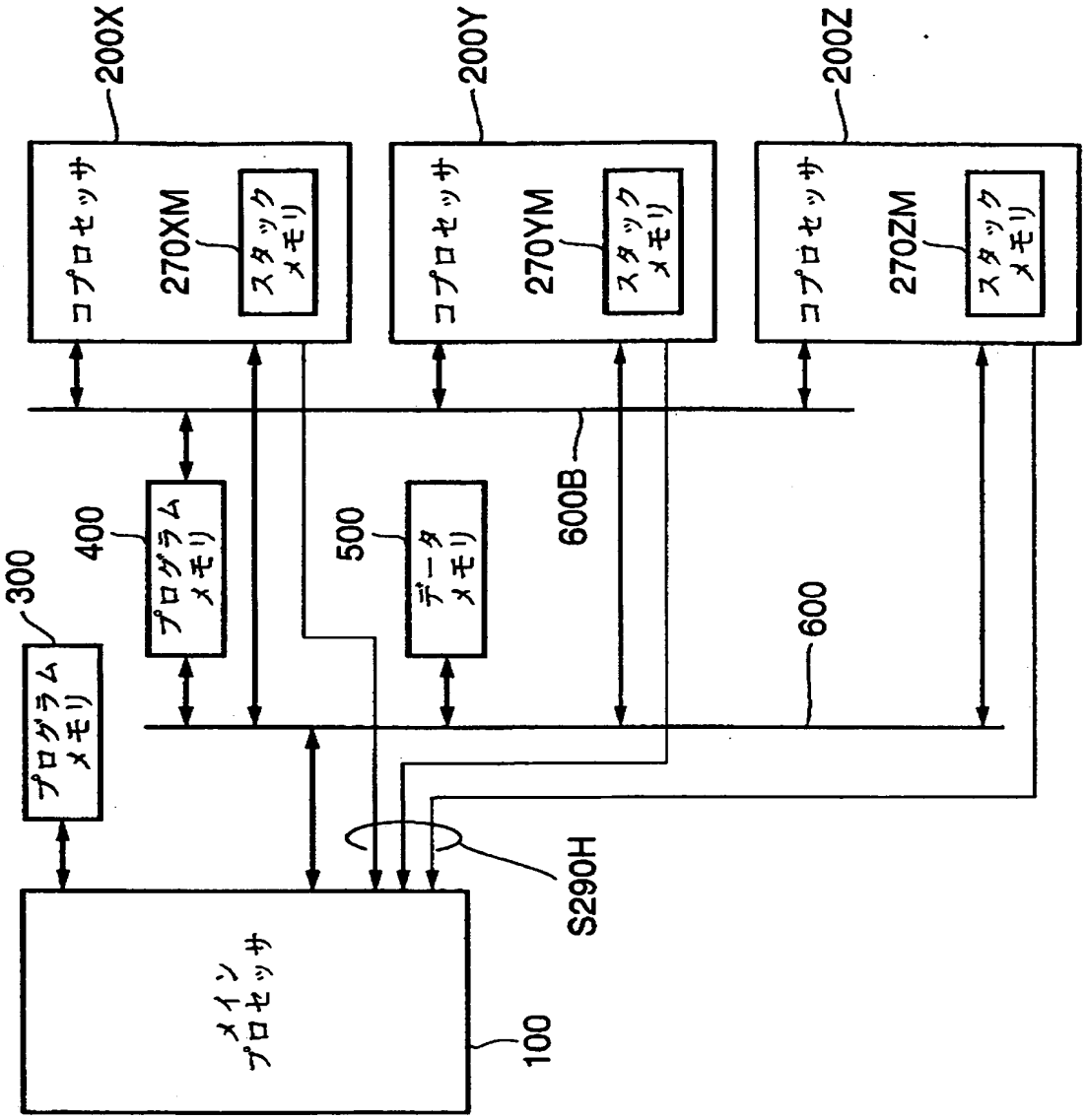
【図１０】


アドレス
0x1050
0x104C
SP
0x1048
0x1044
0x1014
0x1010
コンテキストm
コンテキストm-1
d27-n
d27-2
d27-1
D27
D26
D25
D24
D23
D22
D21
コンテキスト(新)
ローカル変数(新)
引数データ(新)
演算データ(旧)
コンテキスト(旧)
ローカル変数(旧)
引数データ(旧)
(フレーム変更後)
引数データn
引数データn-1
d25-m
d25-2
d25-1
D25
D24
D23
D22
D21
アドレス
0x1014
0x1010
SP
0x100C
0x1008
引数データ(新)
演算データ
コンテキスト
ローカル変数
引数データ(旧)
(フレーム変更前)

【図１１】


200B
(コプロセッサ)
201
210
制御レジスタ
220
命令フェッチ回路
230
命令キュー
240
命令デコーダ
250A
250B
加減算器
加算器
SP
PC
260A
260B
270A
キャッシュメモリ
270C
スタックトップレジスタ
ALU
280
290
割り込み要求発生回路
300
プログラムメモリ(2)
400
プログラムメモリ(1)
500
データメモリ
270B
スタックメモリ
600
100
110
(メインプロセッサ)
120
命令フェッチ回路
130
命令キュー
140
命令デコーダ
150
加算器
160
PC
170
汎用レジスタ
180
ALU
190
データアクセス回路
割り込み要求受付回路

【図１２】


200X
200Y
200Z
コプロセッサ
270XM
スタックメモリ
コプロセッサ
270YM
スタックメモリ
コプロセッサ
270ZM
スタックメモリ
600B
400
プログラムメモリ
500
データメモリ
600
300
プログラムメモリ
S290H
メインプロセッサ
100

【書類名】　　要約書

【要約】

【課題】　回路規模の増大を抑えつつ、特定の命令セットを高速に実行することが可能なマイクロプロセッサシステムを提供すること。

【解決手段】　プログラムに記述された命令を実行するマイクロプロセッサシステムにおいて、第１の命令セットをハードウェア上で実行すると共に第２の命令セットをソフトウェア上で実行するメインプロセッサ１００と、前記メインプロセッサの管理下で動作して前記第２の命令セットをハードウェア上で実行するコプロセッサ２００とを備える。前記コプロセッサ２００は、前記第２の命令セットのうち、前記メインプロセッサ１００の管理下にあるデータを操作する必要を生ずる特定の命令に遭遇した場合、前記メインプロセッサ１００に通知を発行して当該命令の実行を依頼する。この場合、コプロセッサ２００は、ハードウェア上でスタックポインタおよびプログラムカウンタを自ら更新する。

【選択図】　　図１

認定・付加情報

特許出願の番号	平成１１年　特許願　第３３１９６０号
受付番号	５９９０１１４０６７１
書類名	特許願
担当官	坏　政光　　８８４４
作成日	平成１１年１１月２９日

＜認定情報・付加情報＞

【特許出願人】

【識別番号】　000004237

【住所又は居所】　東京都港区芝五丁目７番１号

【氏名又は名称】　日本電気株式会社

【代理人】　申請人

【識別番号】　100108578

【住所又は居所】　東京都新宿区高田馬場３丁目２３番３号　ＯＲビル　志賀国際特許事務所

【氏名又は名称】　高橋　詔男

【代理人】

【識別番号】　100064908

【住所又は居所】　東京都新宿区高田馬場３丁目２３番３号　ＯＲビル　志賀国際特許事務所

【氏名又は名称】　志賀　正武

【選任した代理人】

【識別番号】　100101465

【住所又は居所】　東京都新宿区高田馬場３丁目２３番３号　ＯＲビル　志賀国際特許事務所

【氏名又は名称】　青山　正和

【選任した代理人】

【識別番号】　100108453

【住所又は居所】　東京都新宿区高田馬場３丁目２３番３号　ＯＲビル　志賀国際特許事務所

【氏名又は名称】　村山　靖彦

次頁無

出　願　人　履　歴　情　報

識別番号　　〔０００００４２３７〕

１．変更年月日　　１９９０年　８月２９日
［変更理由］　新規登録
住　所　東京都港区芝五丁目７番１号
氏　名　日本電気株式会社